



# A New Simplified Approach to Federated Single Sign-on System

By

Chen Liang B.Sc.

Supervisor: Dr. Fredrick Mtenzi

Supervisor: Mr. Paul Doyle

Thesis submitted to the Office of Postgraduate Studies and Research at the Dublin  
Institute of Technology in fulfilment of the requirements for the degree of Master of  
Philosophy (M.Phil.)

School of Computing

Dublin Institute of Technology

Kevin Street, Dublin 8, Ireland

# Abstract

Currently, it is not possible to provide a single sign-on session (based on a federated authentication model) for end-users to access diverse computer services/applications (in multiple domains) without requiring re-authentication. Network domains provide diverse computer services/applications that range from web-based email services to remote desktop applications. To access these divers computer services/applications, end-users are required to proof their identities (authentication). The more computer services/applications the end-users access, the more authentication processes the end-users have to go through. Federated single sign-on (FSSO) uses a claim based authentication approach that aims to allow end-users to access multiple domains within a single authentication session. However, federated single sign-on has been largely employed in web space. When end-users accessing diverse (web-based and non-web based) computer services/applications, they have to authenticate multiple times.

There are currently four separate federated single sign-on projects aiming to address this limitation: Project Moonshot, SASL-SAML (i.e. the combination Simple Authentication and Security Layer with security assertion markup language), Kerberos Secure Sharing (KSS) and SAML-AAI/Kerberos (i.e. the combination of Security Assertion Markup Language's authentication authorisation infrastructure with Kerberos single sign-on infrastructure). Project Moonshot aims to create a separate federated single sign-on systems for non-web based applications (i.e. two distinct authentication processes for both web-based and non-web based applications, so this is not strictly FSSO). KSS aims to extend Kerberos beyond a single network domain to support federated single sign-on. While it can work in conjunction with web-based authentication to provide a true single sign-on experience, if network domains are unwilling to expose their Kerberos services, this model will not be accepted by network domains. SASL-SAML and SAML-AAI/Kerberos aim to combine web-based federated single sign-on systems with non-web based applications. SASL-SAML modifies non-web based service applications and their user agents so that they can use web-based federated single sign-on systems and tokens. These modifications

require active support and buy-in from each application provider and network domains without which the model will fail. SAML-AAI/Kerberos modifies the SAML based service providers so they can access non-web based services/applications on behalf of the end-users. The dependence of web-based authentication means SAML-AAI/Kerberos requires the use of a web-browser (with or without graphic user interface) for federated single sign-on. Application providers need to create web-based user agents for their services otherwise this model will fail. In addition, end-users need to authenticate to their desktop to access web-browsers. Two authentication processes are required by this model.

The presented approaches develop new authentication systems that enable federated single sign-on in web and non-web space. Unfortunately network domains are reluctant to modify their existing authentication infrastructures and existing computer services/applications to support these approaches. The aim of this research was to design a new simplified federated single sign-on system: a novel federated single sign-on system that incorporates existing authentication infrastructures, and testing environment. By incorporating the existing authentication infrastructure, this system is simpler to deploy and more acceptable to network domains than the existing approaches. The characteristics of the existing approaches were examined, with the aim of determining the components where authentication system may be extended. The findings have been incorporated into the design of the system.

# Declaration

I certify that this thesis which I now submit for examination for the award of *Master of Philosophy* is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate study by research of the Dublin Institute of Technology and has not been submitted in whole or in part for another award in any other third level institution.

The work reported on in this thesis conforms to the principles and requirements of the DIT's guidelines for ethics in research.

DIT has permission to keep, lend or copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Signature \_\_\_\_\_ Date \_\_\_\_\_

Candidate

# Acknowledgements

I owe sincere and earnest thankfulness to my supervisors, Dr. Fredrick Mtenzi and Mr. Paul Doyle, for the support and guidance he showed me throughout my research and writing. In addition to direct my research, they advised me on how to overcome the difficulties in the process of a research.

I also would like to thanks Prof. Brendan O'Shea and Mr. Mark Deegan. This dissertation would not have been possible unless they helped me to secure my position as a research student in School of Computing of Dublin Institute of Technology.

In the end, I would like to show my gratitude to my parents for their support. They provided continue financial support that allowed me to achieved my goal.

**Chen Liang**

# Table of Contents

<b>Abstract</b>	<b>ii</b>
<b>Declaration</b>	<b>iv</b>
<b>Chapter 1: Introduction</b>	<b>1</b>
1.1 Research Background . . . . .	4
1.1.1 Authentication Models . . . . .	4
1.1.2 Federated Single Sign-On . . . . .	5
1.2 Research Aim and Objectives . . . . .	6
1.3 Contributions . . . . .	7
1.3.1 Primary Contribution . . . . .	7
1.3.2 Secondary Contribution . . . . .	7
1.4 Thesis Structure . . . . .	8
<b>Chapter 2: Basic Concepts</b>	<b>10</b>
2.1 Introduction . . . . .	10
2.2 A Brief History of Computer Systems and Security . . . . .	11
2.3 Identity Management . . . . .	12
2.3.1 Digital Identity . . . . .	14
2.3.2 Authentication . . . . .	15
2.3.3 Authorisation . . . . .	15
2.3.4 Auditing . . . . .	16
2.4 Summary . . . . .	16
<b>Chapter 3: Authentication Systems</b>	<b>17</b>

3.1	Introduction . . . . .	17
3.2	Authentication Mechanisms . . . . .	18
3.2.1	Username and Password . . . . .	19
3.2.2	Challenge and Response . . . . .	20
3.2.3	Digital Certificates . . . . .	20
3.2.4	Biometrics . . . . .	21
3.3	Authentication Models . . . . .	23
3.3.1	Silo Model . . . . .	23
3.3.2	Centralised Identity Management (CIM) . . . . .	25
3.3.3	Centralised Single Sign-on . . . . .	27
3.3.4	Federated Single Sign-on . . . . .	30
3.4	Summary . . . . .	33
<b>Chapter 4: Federated Single Sign-On (FSSO) Systems</b>		<b>34</b>
4.1	Introduction . . . . .	34
4.2	Federated Single Sign-On Solutions in Web Space . . . . .	38
4.2.1	Security Assertion Markup Language (SAML) V1.0 . . . . .	39
4.2.2	Liberty Identity Federation Framework (ID-FF) . . . . .	41
4.2.3	WS-Federated Authentication Module . . . . .	42
4.2.4	Security Assertion Markup Language (SAML) V2.0 . . . . .	44
4.3	Federated Single Sign-On Solutions Beyond Web Space . . . . .	45
4.3.1	Project Moonshot . . . . .	45
4.3.2	Kerberos Secure Sharing . . . . .	48
4.4	Federated Single Sign-On Approaches in Both Space . . . . .	50
4.4.1	SASL-SAML . . . . .	50
4.4.2	SAML-AAI/Kerberos . . . . .	51
4.5	Summary . . . . .	54
<b>Chapter 5: The Design of a New Simplified FSSO System</b>		<b>55</b>
5.1	Introduction . . . . .	55

5.2	Research Assumptions . . . . .	56
5.3	Simulation of Real World Applications . . . . .	57
5.4	Goals of a Simplified Federated Single Sign-On System . . . . .	57
5.5	Environment Specification . . . . .	57
5.6	Requirements Specification . . . . .	59
5.7	Use Cases . . . . .	59
5.7.1	Use Case 1 . . . . .	60
5.7.2	Use Case 2 . . . . .	60
5.7.3	Use Case 3 . . . . .	61
5.8	Simplified Federated Single Sign-on System Life-cycle Description . . . . .	61
5.8.1	Life-cycle 1: End-user accesses non-web base application . . . . .	62
5.8.2	Life-cycle 2: End-user accesses web-based application . . . . .	64
5.8.3	Other Life-cycles: End-user accesses web-based and non-web based applications . . . . .	65
5.9	New Simplified Federated Single Sign-on System Components . . . . .	67
5.9.1	Network Domain Design . . . . .	67
5.9.2	End-user Database Design . . . . .	68
5.9.3	Federated Single Sign-On Policy Design . . . . .	71
5.9.4	Kerberos Single Sign-On Simulation Design . . . . .	72
5.9.5	Non-web Based Service Application Simulation Design . . . . .	80
5.9.6	SAML based FSSO Simulation Design (Identity Provider) . . . . .	89
5.9.7	SAML based FSSO Simulation Design (Service Provider) . . . . .	94
5.9.8	Middle-ware System Design . . . . .	95
5.10	Summary . . . . .	100
<b>Chapter 6: Implementation of the New Simplified FSSO System</b>		<b>101</b>
6.1	Introduction . . . . .	101
6.2	Technical Overview . . . . .	102
6.2.1	Development Environment and Programming Languages . . . . .	102
6.2.2	System Configuration . . . . .	102



6.3	The Simplified Federated Single Sign-On System User Manual . . . . .	103
6.3.1	Kerberos Single Sign-on Server Simulation . . . . .	103
6.3.2	SAML based Federated Single Sign-On Simulation . . . . .	110
6.3.3	Accessing Non-web based Service in Another domain . . . . .	118
6.4	Summary . . . . .	127
<b>Chapter 7: Research Evaluation</b>		<b>128</b>
7.1	Introduction . . . . .	128
7.2	Criteria Development . . . . .	128
7.3	Evaluation using the Developed Criteria . . . . .	129
7.3.1	First Criterion . . . . .	129
7.3.2	Second Criterion . . . . .	131
7.3.3	Third Criterion . . . . .	132
7.3.4	Fourth Criterion . . . . .	133
7.3.5	Fifth Criterion . . . . .	134
7.3.6	Sixth Criterion . . . . .	136
7.3.7	Seven Criterion . . . . .	137
7.4	Comparison Overview . . . . .	137
7.5	Summary . . . . .	139
<b>Chapter 8: Discussion and Conclusion</b>		<b>140</b>
8.1	Introduction . . . . .	140
8.2	Contributions . . . . .	141
8.2.1	Primary Contribution . . . . .	141
8.2.2	Secondary Contributions . . . . .	142
8.3	Future Work . . . . .	142
8.4	Potential Application of the new Simplified FSSO System . . . . .	144
8.4.1	Federated Single Sign-On in Cloud Computing . . . . .	144
<b>References</b>		<b>145</b>

<b>Appendices</b>	<b>156</b>
<b>Chapter A: System Specifications</b>	<b>156</b>
A.1 Hardware Specifications . . . . .	156
A.2 Software Specifications . . . . .	156
<b>Chapter B: List of Security Technologies</b>	<b>159</b>
B.1 The Simple Authentication and Security Layer (SASL) . . . . .	159
B.2 Generic Security Service Application Program Interface (GSS-API) . . . . .	160
B.3 Extensible Authentication Protocol (EAP) . . . . .	161
B.4 Lightweight Directory Access Protocol (LDAP) . . . . .	161
B.5 ActiveDirectory . . . . .	162

# List of Figures

3.1	Silo Model (Jøsang et al., 2007) . . . . .	24
3.2	An end-user must store credential in every domain . . . . .	25
3.3	Centralised Management Solution (Jøsang et al., 2007) . . . . .	26
3.4	Federated Single Sign-On (Jøsang et al., 2007) . . . . .	31
3.5	Federated Single Sign-On Process . . . . .	32
4.1	Federated Single Sign-On (Jøsang et al., 2007) . . . . .	35
4.2	The SAML Domain Model . . . . .	40
4.3	SAML 2.0 incorporates three existing federation technologies and standards – SAML 1.1, ID-FF 1.2 and Shibboleth (Jason & Goode, 2012). . . . .	44
4.4	Project Moonshot Model (Howlett, 2011) . . . . .	47
4.5	Kerberos Secure Sharing (Gridwise Tech, 2010) . . . . .	49
4.6	SAML-SASL Model . . . . .	51
4.7	Interoperating SAML-AAI and Kerberos (Papez, 2009) . . . . .	52
5.1	A New Simplified Federated Single Sign-on Life-cycle 1. . . . .	64
5.2	A New Simplified Federated Single Sign-on Life-cycle 2. . . . .	66
5.3	End-user Database Written in XML. . . . .	71
5.4	Federation Policies Written in XML. . . . .	72
5.5	Database Tree Relation. . . . .	75
5.6	Database Tree Relation in Colour. . . . .	76
5.7	Simulate Kerberos Ticket Granting Ticket. . . . .	77
5.8	Sample Service Database Written in XML. . . . .	78
5.9	Sample Source String. . . . .	79
5.10	Sample Shared Key. . . . .	79

5.11 Sample Service Ticket. . . . .	80
5.12 Unix/Linux Shell and SSH simulation. . . . .	81
5.13 Sample SSH server simulation Entry in the Service Database. . . . .	87
5.14 Sample Shared Key between SSH Server and Kerberos Server Simulation. .	88
5.15 Sample Message that includes a Username and Service Ticket. . . . .	88
5.16 Username/Password Portal Design. . . . .	90
5.17 Sample Identity Provider Entry in the Service Database. . . . .	92
5.18 Sample Shared Key between Identity Provider Simulation and Kerberos Server Simulation. . . . .	92
5.19 Kerberos Ticket Granting Ticket Portal Design. . . . .	92
5.20 Sample Identity Provider Entry in the Service Database. . . . .	96
5.21 Sample Shared Key of Middle-ware system's Identity Provider. . . . .	97
5.22 Sample Identity Assertion. . . . .	98
5.23 Sample Ticket Granting Ticket of Domain Beta. . . . .	99
6.1 Start KDC Simulation in domain home.virtual.vm. . . . .	104
6.2 Remote Unix/Linux desktop simulation requests shared key from KDC simulation. . . . .	104
6.3 End-user login the local Unix/Linux desktop and receives ticket granting ticket. . . . .	105
6.4 Command, "klist", display the end-user's ticket granting ticket; command, "hostname" displays the hostname of the current machine; command, "if- config" displays the local IP address of the current machine. Command, "ls", display all the files in current directory. . . . .	107
6.5 End-user login to the remote Unix/Linux desktop with secure shell. . . . .	108
6.6 End-user executes commands on the remote Unix/Linux desktop. . . . .	109
6.7 Start KDC in domain foreign.virtual.vm. . . . .	110
6.8 Remote Unix/Linux desktop simulation in domain foreign.virtual.vm. . . .	111
6.9 Access Web-based Service Provider in domain foreign.virtual.vm. . . . .	112
6.10 Use Kerberos Authentication Method in domain home.virtual.vm. . . . .	113

6.11	The end-user is successfully authenticated. . . . .	114
6.12	SAML Based Federated Single Sign-On Simulation in domain foreign.virtual.vm .	115
6.13	We-based interface for accessing remote Unix/Linux desktop simulation. .	116
6.14	Executes sample Unix/Linux commands. . . . .	117
6.15	Start KDC simulation in domain home.virtual.vm and foreign.virtual.vm .	119
6.16	Start middle-ware system (identity provider) in domain home.virtual.vm .	120
6.17	End-user login the Unix/Linux desktop simulation and receives a ticket granting ticket in home.virtual.vm . . . . .	121
6.18	The remote Unix/Linux desktop simulation in domain foreign.virtual.vm receives a shared key . . . . .	122
6.19	Start the middle-ware system (service provider) in domain foreign.virtual.vm . . . . .	123
6.20	End-user attempts to access the Unix/Linux desktop in foreign.virtual.vm .	123
6.21	KDC simulation in domain home.virtual.vm verifies the end-user ticket granting ticket and issues a service ticket . . . . .	124
6.22	Middle-ware system (identity provider) in domain home.virtual.vm verifies the end-user's identity and sends an identity assertion to the middle-ware system (service provide) in domain foreign.virtual.vm . . . . .	124
6.23	Middle-ware system (service provider) in domain foreign.virtual.vm accepts the end-user's identity assertion. It then requests ticket granting ticket and service ticket on behalf of the end-user in domain foreign.virtual.vm. . . .	124
6.24	KDC simulation in domain foreign.virtual.vm issues ticket granting ticket and service ticket. The service ticket is passed to the end-user in domain home.virtual.vm through the middle-ware system . . . . .	125
6.25	The remote Unix/Linux desktop simulation in domain foreign.virtual.vm accepts the request from the end-user. . . . .	125
6.26	The remote Unix/Linux desktop simulation in domain foreign.virtual.vm accepts the request from the end-user. The end-user performed a file cre- ation task. . . . .	126

A.1	Dell Optiplex GX260 configuration. . . . .	158
B.1	SASL is conceptually a framework that provides an abstraction layer between protocols and mechanisms(Melnikov & Zeilenga, 2006) . . . . .	160

# List of Tables

3.1	Protocols that Already Support Kerberos (MIT, 2008)	30
4.1	Difference Between Silo Model, Centralised Identity Management Model and Federated Single Sign-On Model	36
4.2	Authority-centric and User-centric Federated Single Sign-On	37
5.1	Network Domain Names	67
5.2	Domain Variables	70
7.1	Comparing approaches with the first criterion	130
7.2	Comparing approaches with the second criterion	132
7.3	Comparing approaches with the third criterion	133
7.4	Comparing approaches with the fourth criterion	134
7.5	Comparing approaches with the fifth criterion	135
7.6	Comparing approaches with the sixth criterion.	136
7.7	Comparing approaches with the seventh criterion.	137
7.8	Table comparison	139
A.1	IBM eServer Hardware Overview.	157
A.2	Sun Ultra 40M2 Workstation Configuration.	157
A.3	HP Probook 6550b Software Configuration.	158

# Chapter 1

## Introduction

Currently, it is not possible for an end-user to access diverse computer services (web-based and non-web based) in multiple domains with a single authentication session. To access computer services that are provided by multiple network domains, end-users need to authenticate multiple times. This results in time lost on repeating authentications. The increasing amount of authentication process also results in password fatigue (i.e. end-users are required to remember an excessive number of passwords). It also causes end-users to enter their credentials at any authentication request without identifying the request party, which leaves them vulnerable to phishing attacks.

Federated single sign-on (FSSO) model attempts to address this issue with claim-based authentication. It establishes a federation (i.e. a circle of trust) between network domains (Wason et al., 2003). In a federation, end-users authenticate to their network domains' authentication systems and acquire identity assertions. These identity assertions claim that the end-users have been authenticated by parties within the federation. With the identity assertions, the end-users can access computer services/applications in any domains (within the federation) without re-authentication.

Federated single sign-on standard – Security Assertion Markup Language (SAML) (OASIS, 2010; Madsen et al., 2005) – aims to deliver a single sign-on experience for accessing web-based computer services/applications. It can be incorporated into network



domains' web-based authentication systems. It has been used by major academic networks (e.g. HEAnet) to establish federation between universities and institutes (e.g. DIT, Trinity, UCD). It allows students of these parties to access web-based resources (within the federation) with one authentication session.

SAML, however, only partially addresses the single sign-on issue. It does not address federated single sign-on in non-web space. Network domains provide diverse computer services/applications that range from email services to remote desktop applications. With SAML, an end-user can access web-based services/applications with a single authentication session; however, the same session can not be used for non-web base services/applications.

HEAnet have commenced a pilot project to deliver a federated access infrastructure to the research and education sector within Ireland (HEAnet, 2008). According to HEAnet (2012a), the pilot project has successfully established a federation between major universities and institutes (e.g. Dublin Institute of Technology, Dublin City University, UCD Connect Bridge, University of Limerick). So far, the pilot project uses SAML based federated single sign-on system, therefore, staff and students can access web-based services with one federated authentication session. According to Glenn Wearen (the Middleware Specialist in HEAnet), enabling federated single sign-on for diverse computer services/applications (web-based and non-web based) are still a major obstacle.

There are four project attempt to address this issue, Project Moonshot (Howlett, 2011), Kerberos Secure Sharing (KSS) (Gridwise Tech, 2010), SASL-SAML (Wierenga & Lear, 2010) (i.e. the combination Simple Authentication and Security Layer with security assertion markup language) and SAML-AAI/Kerberos (Papez, 2009) (i.e. the combination of Security Assertion Markup Languages authentication authorisation infrastructure with Kerberos single sign-on infrastructure). Projects such as Project Moonshot, Kerberos Secure Sharing aim to develop federated single sign-on systems for non-web based services; while projects such as SASL-SAML and SAML-AAI/Kerberos aims to develop federated single sign-on systems for both space.

Similar to SAML, Project Moonshot (Howlett, 2011) and KSS (Gridwise Tech, 2010) only partially address the single sign-on issue. In contrast to SAML, Project Moonshot and KSS focus on accessing non-web based applications with one federated single sign-on session. While they can be used in conjunction with SAML, end-users still need to authentication multiple times to access diver computer services/applications.

SASL-SAML (Wierenga & Lear, 2010) and SAML-AAI/Kerberos (Papez, 2009) aim to extend SAML based federated single sign-on system to non-web based services/applications. By modifying network domains' authentication infrastructure and their non-web based services/applications to support these federated single sign-on systems, these systems allows end-users to access both web-based and non-web based services/applications with one authentication session. The modifications to computer services/applications, however, require active support and buy-in from application providers without which the model will fail (i.e. they will not be adopted by network domain if network domains and application providers were reluctant to modify their existing services/applications).

These is a need for a new simplified federated single sign-on system. This system should support both web-based and non-web based services/applications (i.e. similar to SASL-SAML and SAML-AAI/Kerberos). More importantly, it should be simple to deploy in network domains: it should minimise the modification to existing computer services/applications, and it should leverage the existing authentication infrastructure of network domains. This way, this system is more acceptable by network domains.

Section 1.1 presents the research background; it gives a brief description of the history and existing approaches of federated single sign-on. Section 1.2 presents the aims and objectives of this research. Section 1.3 lists this research's contributions. Section 1.4 presents the structure of the thesis.

## 1.1 Research Background

### 1.1.1 Authentication Models

Federated single sign-on is one of the authentication models. Before presenting the background of federated single sign-on systems, it is important to introduce authentication model and present the other authentication models (e.g. silo model and centralised identity management model). This will help us to understand the history of federated single sign-on model.

An authentication model has two components, identity provider and service provider. An identity provider manages the identities of end-users. It authenticates the end-users, and it provides identity assertion to authenticated end-users. Service provider provides computer services to authenticated end-users. It does not provide any authentication by itself.

In silo model, identity provider and service provider reside in the same space; each domain has an identity provider and service provider (Jøsang & Pope, 2005; Jøsang et al., 2007; Vacca et al., 2010). In a domain that uses silo model, its identity provider authenticates end-user for its service provider. For an end-user to access computer services in a domain, the end-user needs to be authenticated by that domain's identity provider. Silo model does not provide any cross domain authentication, therefore, multiple authentication sessions are required for an end-user to access computer services in multiple domains.

To address the cross domain authentication issues, *centralised identity management model* uses a centralised identity provider approach. It removes the identity providers in all of the domains, instead, it uses a single centralised identity provider to manage all the identities for every domain (Jøsang et al., 2007). This approach allows end-users to access computer services in multiple domains by authenticating to the single identity provider. This approach, however, presents trust issues; Chappell (2006) argues that no single party can be trusted to manage all the identities in the world.

### 1.1.2 Federated Single Sign-On

Unlike centralised identity management approach, federated single sign-on proposes a decentralised identity provider approach. Federated single sign-on proposes domains to manage their own identity providers. It builds a circle of trust (i.e. federation) between domains. In the federation, end-users access computer services in multiple domains by claiming they are from one of the federation's domain. For example, an end-user has been authenticated by a domain's identity provider, the identity provider can vouch for the end-user when he/she accesses service providers of the other domains. Since these domains are in a federation, the service providers of the other domains will accept the requests from the end-user.

Federated single sign-on was developed for web-based cross domain single sign-on (Maler et al., 2003; Sun Microsystems, 2004; Shim et al., 2005; Internet2, 2012). Security Assertion Markup Language (SAML) (OASIS, 2010; Madsen et al., 2005) was developed by Organization for the Advancement of Structured Information Standards (OASIS, 2011) for this purpose. SAML based software, Shibboleth (Internet2, 2012), was implemented. Liberty alliance developed the Liberty Identity Web Services Framework for web-based federated single sign-on. Later Liberty alliance and Shibboleth contributed their work into SAML 2.0. SAML 2.0 currently is the *de facto* standard for web-based federated single sign-on (Ragouzis et al., 2008).

Even though, the concept of federated single sign-on was largely explore in web space (Wierenga & Lear, 2010; Howlett, 2011). These is a need to explore the same concept in non-web space. There are four approaches aiming to do that, Project Moonshot, Kerberos Secure Sharing, SASL-SAML and SAML-AAI/Kerberos. Approach such as *Project Moonshot* (Howlett, 2011) modifies the non-web based services/applications to support federated single sign-on. *Kerberos secure sharing (KSS)* (Gridwise Tech, 2010) uses Kerberos' cross realm authentication to enable federated single sign-on for non-web based applications. Project Moonshot and KSS, however, do not explore federated single sign-on in web and non-web space. Therefore, if an end-user accesses computer services

in both space, two separate authentication processes are needed. In addition, these approaches will fail if network domains are reluctant to modify their existing authentication infrastructures and services/applications to support these approaches (Painless Security, 2010).

Unlike Project Moonshot and KSS, *SASL-SAML* (Wierenga & Lear, 2010) and *SAML-AAI/Kerberos* (Papez, 2009) explore the concept in both web and non-web space. Although they can provide a single authentication session for end-users to access both web-based and non-web based services, they also require network domains' services/applications to be modified. Similar to Project Moonshot, if the provider of these services/applications are unwilling to modify their existing applications, these approaches will fail.

## 1.2 Research Aim and Objectives

This research aims to develop a new simplified federated single sign-on approach. This approach assumes that network domains use their own Kerberos single sign-on system for single sign-on, and a federation has been established between these domains using SAML based federated single sign-on system.

This approach aims to allow end-users to access diverse computer services in multiple domains with a single authentication session. In addition, it aims to present minimum modification to existing services/applications, and it aims to leverage the network domains' existing authentication infrastructures. Therefore, it increases the acceptance of the system by network domains. This research draw extensively on OASIS's work on Security Assertion Markup Language (SAML) (OASIS, 2010), Rok Papez's SAML-AAI/Kerberos approach (Papez, 2009) and MIT's work on Kerberos single sign-on (Miller et al., 1988).

The research presented in this thesis is validated in experiments using simulations of real world environments. The experiment shows that the prototype system can incorporate the network domains' existing authentication infrastructure (in contrast to

modifying them). It however, needs to add extra functions to the non-web based services/applications. The extra functions enable non-web based services/applications to use different authentication types (i.e. local network authentication or federated authentication). While this approach hasn't eliminates the needs to modify existing computer services/applications, it had made progress towards simplifying the deployment of federated single sign-on.

## **1.3 Contributions**

### **1.3.1 Primary Contribution**

The development of a new simplified federated single sign-on system represents an important contribution to the study of federated single sign-on. This work is a novel attempt to develop a federated single sign-on system which supports both web-based and non-web based services/applications without modifying them. This approach is developed based on the research of existing approaches (e.g. Project Moonshot, SASL-SAML, SAML-AAI/Kerberos and Kerberos Secure Sharing). Assuming that network domains use their own Kerberos authentication systems and a federation has been established between them, this approach allows end-users to access diverse computer services in multiple domains with a single authentication session. In addition, this approach is simpler to adopt by network domains than existing approaches.

### **1.3.2 Secondary Contribution**

#### **Secondary Contribution 1**

The research explores the concept of federated single sign-on in web and non-web space. It critically reviewed the existing approaches in literature, and it developed a understanding of federated single sign-on systems. The research findings have contributed to the field of federated single sign-on research.

#### **Secondary Contribution 2**

The research develops a testing environment that includes a simulations of Kerberos single sign-on system, SAML based federated single sign-on system, Secure shell protocol and Unix/Linux based desktop system. These simulations are used to evaluate the research prototype.

## 1.4 Thesis Structure

Chapter 2, Basic Concepts, presents the basic concepts of computer applications. It presents the development of computer systems and their effects on the development of computer security. In addition, it describes the components – digital identity, authentication, authorisation and auditing – of identity management.

Chapter 3, Authentication Systems, explores deeper into computer security. It presents the definition, history and basic concepts of authentication systems. The chapter includes the evolution of authentication mechanisms and authentication models.

Chapter 4, Federated Single Sign-On (FSSO) Systems, focuses on one of the authentication models, federated single sign-on. It presents the definition and basic concept of federated single sign-on. It also compares the existing approaches in literature and gives a comprehensive review of federated single sign-on approaches. The federated single sign-on approaches are presented and evaluated in a table. The issues of existing approaches are identified and the need of a new simplified federated single sign-on system is proposed.

Chapter 5, Simplified FSSO System Requirements, presents the requirement specification for designing a new simplified federated single sign-on system. It describes the goal of the new simplified federated single sign-on system; it also specifies the details of the design environment such as network domains, computer services and communication protocols.

Chapter 6, Simplified FSSO System Design, describes the design of the new simplified federated single sign-on system. It includes designing the simulations of Kerberos

single sign-on, SAML based single sign-on, Secure shell protocol, Unix/Linux desktop and a web-service. It also includes the design of a middle-ware system that allows end-users to receive and use Kerberos service ticket from another domain.

Chapter 7, Implementation of the New Simplified FSSO System, describes the implementation of the new simplified federated single sign-on system. It presents the set-ups of hardware and software. In addition, it presents the end-user manual of the system.

Chapter 8, Research Evaluation, presents the evaluation of the new simplified federated single sign-on system. It develops six comparison criteria and uses them to compare the new simplified FSSO system with Project Moonshot, KSS, SASL-SAML, SAML-AAI/Kerberos.

Chapter 9, Discussion and Conclusion, presents the discussion and conclusion of the research.



# Chapter 2

## Basic Concepts

### 2.1 Introduction

The main focus of this research is federated single sign-on (FSSO) of digital identity. However, before presenting the literature review on FSSO, it is necessary to present the basic concepts and the context of the research. Therefore, this chapter presents a brief and general background on computer, computer security and identity management.

To help reader to understand the terms in this research, an entity is a generic term that refers to an active agent capable of initiating or performing a computation of some sort (Benantar, 2005). An entity can be human or non-human. A non-human entity can be a computer device that capable of performing a computational task. In this research, the terms “end-user” and “entity” are referred to human entities. The term “organisation” is refers to an organized body of people with a particular purpose (e.g. a business, government department, charity).

Section 2.2 presents a brief history of computer systems and their respective security challenges. Section 2.3 introduces identity management, identity management’s functionalities and its place in computer security. In section 2.3, there are four subsections. Subsection 2.3.1 introduces digital identity, which is an essential part of identity management. Subsection 2.3.2 2.3.3 and 2.3.4 present the background information of

Authentication, Authorisation and Auditing (AAA). Section 2.4 summaries this chapter.

## 2.2 A Brief History of Computer Systems and Security

In the beginning, computer devices (Oxford English Dictionary, 2008) are combinations of electronic circles and switches Hedin (1937); Pfleeger & Pfleeger (2006). To secure such devices, one can only use physical means. These devices (e.g. Colossus (Hinsley & Stripp, 2001)) were secured by placing them at the location that inaccessible to the others.

The introduction of operating system (Kilburn et al., 1961; Oxford English Dictionary, 2004), compatible time sharing system (Corbato et al., 1963, 1992; Walden & Vleck, 2011), computer software (Oxford English Dictionary, 2011) and computer network (Licklider et al., 1968; Cerf et al., 1974) presents new security challenges. Not only the physical machine needs to be secured, there is also a need to secure software systems (from local or remote access) (MacKenzie & Pottinger, 1997). In addition, computer security needs to address the time-sharing nature of operating systems. One end-user's action should not have any negative effects on the others.

According to (Saltzer & Schroeder, 1975), security specialists (e.g. Anderson (1973)) place potential security violations in three categories:

- 1 Unauthorised information release: an unauthorised end-user is able to read and take advantage of information stored in the computer.
- 2 Unauthorised information modification: an unauthorised person is able to make changes in stored information.
- 3 Unauthorised denial of use: an intruder can prevent an authorised end-user from referring to or modifying information even though the intruder may not be able to refer to or modify the information.

In order to avoid the three security violations, three security goals are defined.

The terms Confidentiality, Integrity and Availability have been used to define the three security goals that needs to achieve (Bishop, 2003). Maintaining these three security goals is the main focus of computer security researches. The focus of this research resides in the identity management. The next section (Section 2.3) gives a brief background information on identity management. In addition, it presents the core functionalities of identity management.

## 2.3 Identity Management

The main focus of the research resides in the authentication process of identity management. The more detailed background of authentication is presented in section 2.3.2. This section aims to only give a brief overview of identity management and its core functionalities.

Identity management has been recognized by researcher such as Dunleavy et al. (2006); Halperin & Backhouse (2008) as a key research theme for the coming decades. It is an important part of the computer security. Any organisation that employs a body of people requires identity management. In the age of computing, how to manage identity with computers becomes one of the important research in computing.

Section 2.2 showed that the emergence of operating systems emerged in 1960s presents new security challenges. As a part of computer security, identity management evolves from a physical identity management (human manage human identities) to digital identity management (computers manage human identities). This research researches the management of human identities by digital means, therefore, this research is in the context of the identity management of digital identities.

To defined identity management, this research reviewed the literature of Halperin & Backhouse (2008); Lee et al. (2009); Thomas & Meinel (2009); Balasubramaniam et al. (2009).

- Halperin & Backhouse (2008) refers identity management as “the management of

digital identities or digital identity data.”

- Lee et al. (2009) defines the identity management as “the business process that creates, manages, and uses user IDs, and the infrastructure that supports this process.”
- Thomas & Meinel (2009) refers the identity management as “the establishment and controlled use of identity in the online world.”
- Balasubramaniam et al. (2009) describes identity management as “the tasks needed to create, manage, and decommission user credentials and attributes in a software system.”

The research drew extensively from these literatures and defines identity management as “the processes of creating, managing and decommissioning of end-users’ digital identities.” More detailed background of digital identity will be presented in section 2.3.1.

According to Glasser & Vajihollahi (2008); Bertino & Takahashi (2010), identity management faces these challenges:

- How to ensure the privacy of the digital identities. (Digital identity available only to the right individuals or services at the right time and place).
- How to establish trust between parties involved in the identity transactions.
- How to avoid the abuse of digital identities.
- How to make these provisions possible in a scalable, usable, and cost effective manner.

According to Mont et al. (2002), current trends suggest that the digital world is going to be more and more flexible and dynamic. People are active in a heterogeneous environment. Therefore, many took up multiple roles. While this creates many opportunities in the personal and social business areas. It also raises many new threats and issues.

The core identity management functionalities are Authentication, Authorisation and Auditing (AAA) (Mont et al., 2003). Section 2.3.2, 2.3.3 and 2.3.4 give a brief introduction of the three functionalities. However, the focus of this research resides in the authentication process. Therefore, detailed overviews on authorisation and auditing are out of the scope of this research.

### 2.3.1 Digital Identity

Digital identity is an essential part of identity management. It represents an entity in a specific context (Miyata et al., 2006; El Maliki & Seigneur, 2007). A digital identity consists of traits, attributes, and preferences upon which one may receive personalised services (Project, 2003). According to Vacca et al. (2010), a digital identity was initially considered the equivalent of a user's real-life identity that indicates some of our attributes:

- Who we are: Name, citizenship, birthday.
- What we like: Our favourite reading, food, clothes.
- What our reputation is: Whether we are honest, without any problems.

The research of Windley (2005); Thomas & Meinel (2009) consider digital identities were seen as the electronic extension of an entity's real life identity. It was considered to contain almost the same information as a passport. Seigneur (2005), however, argues the link between the real-world identity and a digital identity is not always mandatory. For example, online video service such as Youtube.com requires the end-users to create identities that are unique from each other. It does not require the end-users to use their real-world identities. It results in many end-users create virtual identities when subscribing to Youtube.com.

According to Benantar (2005), the evolution of computer technologies causes the digital identities to involve from an assigned identifier to an identifier that pointed to various attributes and entitlements. From the last example, Youtube.com involved from an online video sharing service to online video advertisement service. End-users can

receive a revenue share for displaying Youtube.com's advertisements in their personal videos. To receive payments from Youtube.com, end-users must register their personal addresses. Therefore, the personal addresses become an attribute of the end-users' digital identities.

In addition to the increasing complexity of the content of digital identities, the transactions of digital identities are also becoming more and more complex. An end-user may use the same digital identity for multiple computer services from multiple domains. For example, Google Inc. owns services such as gmail and youtube.com. End-user can access youtube.com's contents using their gmail's digital identities.

### **2.3.2 Authentication**

Benantar (2005); Pfleeger & Pfleeger (2006); Vacca et al. (2010) describe authentication as the process of verifying claims about holding specific identities. The process consists of obtaining the authentication information from an entity, analysing the data, and determine if it is associated with that entity (Pfleeger & Pfleeger, 2006). Failure to identify a legitimate end-user will deny the end-user's access to the system. In contract, failure to identify illegitimate end-users will threaten the security of the entire system.

### **2.3.3 Authorisation**

Authorisation is the process of empowering someone to perform an operation or to have access to restricted resources (Salomon, 2005). It manages the level of access of a digital identity. It's the next step after authentication in the identity management functionalities. After an end-user is authenticated, authorisation identifies the level of access of the end-user's digital identity.

### 2.3.4 Auditing

Auditing is the process of collecting and analysing information in order to ensure a proper level of security, as well as compliance with the policies of an organization (Salomon, 2005). It manages (create, modify, and delete) the security policies that are followed by authentication and authorisation. It also manages the provisioning and de-provisioning of the end-users' digital identities.

## 2.4 Summary

Identity management is a part of computer security. It is in charge of creating, managing and decommissioning of end-users' digital identities. It maintains the three security goals of computer systems: confidentiality, integrity and availability. Along with the evolution of computer systems, identity management evolved from physical management (human manages human identity) to digital management (computer manages human identity).

Three core functionalities of identity management are authentication, authorisation and auditing (AAA). Authentication is the main focus of this research. It is the process of verifying claims about holding specific identities. It consists of obtaining the authentication information from an entity, analysing the data, and determine if it is associated with that entity. Next chapter (Chapter 3) will give an in depth review of authentication.

# Chapter 3

## Authentication Systems

### 3.1 Introduction

The focus of this research is the authentication functionality of identity management. As described in section 2.3.2, authentication is the process of verifying claims about holding specific identities (Benantar, 2005; Pfleeger & Pfleeger, 2006; Vacca et al., 2010). It is one of the core functionalities of identity management, the others are authorisation and auditing. According to Pfleeger & Pfleeger (2006), authentications consist of obtaining the authentication information (credential) from an entity, analysing the data, and determining its association with that entity.

Authentication provides the basic security to a computer system. Failure of authentication will threaten the security of the entire system. For example, failure to identify a legitimate end-user will deny the end-user's access to the system, resulting in a denial of service issue. In contrast, failure to identify illegitimate end-users will threaten the confidentiality and integrity of the computer system.

An authentication system contains five elements: the end-user, the distinguishing characteristic of the end-user, the proprietor who is responsible for the system, the authentication mechanism and the access control mechanism (Smith, 2001). The distinguishing characteristics were assigned to the end-user by the proprietor. The authentication mech-



anism authenticates the end-users, and the access control mechanism ensures the level of access that the end-user has.

The evolution of computer systems cause the authentication mechanisms to evolve as well. Physical authentications mechanisms such as guards, spoken passwords and hard-to-forge seals were used to secure a computer system. Due to the emergence of operating system, the authentication system evolved from physical authentication to digital authentication (e.g. username/password, biometrics, one-time password and digital certificate had emerged after username/password) (Smith, 2001).

The rest of the chapter is organised as follows. Section 3.2 presents the authentication mechanisms currently in literature. Section 3.3 presents the evolution of authentication models and the strengths and weaknesses of individual models. In the end, 3.4 summarises the chapter.

## 3.2 Authentication Mechanisms

Many mechanisms such as Username and Password, One-time Password, Digital Certificates and Biometrics exist for authentication (Pfleeger & Pfleeger, 2006; Bishop, 2005). Each authentication mechanism has its strength and weaknesses. According to Pfleeger & Pfleeger (2006); Bishop (2005), authentication mechanisms use any of the following qualities to confirm an end-user's identity:

- Something you know: password, PIN.
- Something you have: one-time-password.
- Something you are: voice, face, fingerprint.
- Where you are: "Location" as another important quality.
- Combination of the four.

Following sections outline the four authentication mechanisms: Username and Password, One-time Password, Digital Certificates and Biometrics. The focus of this research resides

in authentication models, therefore, next sections will present a brief background on authentication mechanisms.

### 3.2.1 Username and Password

Passwords are used as a sole authentication mechanism to a computer-based information system, controlling access to an entire set of computing resources through the operating system. They are mutually agreed-upon code words, assumed to be known only to the user and the system. The passwords whether chosen by users or assigned by the system. The length and format of the password also vary from one system to another.

According to Smith (2001); Pfleeger & Pfleeger (2006), username and password system is widely used, however, it suffers the following issues:

- **Password Guessing**

The strength of the password relies on its complexity. A complex password is difficult to guess but hard to remember or create. If an password is too simple, it opens for guessing or brutal force attack.

- **Lost or forgotten password**

The operators or system administrators often cannot determine what password a user has chosen. If end-users lost their passwords, new ones must be assigned.

- **Redundant passwords**

Supplying multiple passwords for different accesses can be inconvenient and time consuming. It is also difficult for end-users to make multiple passwords, which usually resulting in lost of passwords.

- **Theft of password**

Password is vulnerable to Password File Theft, Keystroke Sniffing and Network Sniffing attacks.

- **Revocation of password**

If one password was shared between multiple end-users to access a common file. Revocation of one end-user's password needs to notify everyone.

### 3.2.2 Challenge and Response

To address the password theft issue, challenge and response uses one-time password scheme. An one-time password (Haller & Metz, 1996) is one that changes every time it is used. Instead of using a static password, the system assigns a static mathematical function to an end-user. The system provides an argument to the function, and the end-user computes and returns the function value. Such systems are also called challenge-response systems because the system presents a challenge to the end-user and judges the authenticity of the end-user by the end-user's response (Pfleeger & Pfleeger, 2006).

Token-based authentication is more secure than authentication mechanism such as username and password. Since it relies on a unique physical object in order to log on, end-user can tell if the token has been stolen. In addition, it is difficult for end-users to share their token with the other and still be able to access the computer system.

Because one-time password requires a client token for calculating password for end-user and a server software for calculating password at the server side. The client token is vulnerable to theft, lost or hardware failure. If a token was lost, the owner of that token will not be able to access the computer systems. If un-authorised end-user stole the token, he/she can attempt to impersonate the owner of that token. In addition, one-time password is more costly than username and password authentication mechanisms in resource and management.

### 3.2.3 Digital Certificates

Digital certificate (or public key certificate) is frequently quoted as the solution critical to authentication of the network. It uses public key infrastructure for authentication. According to Chokhani & Ford (1999), *it binds a public-key value to a set of information*

*that identifies the entity (such as person, organization, account, or site) associated with use of the corresponding private key (this entity is known as the “subject” of the certificate).*

The advantage of digital certificate is that only public key is distributed through the network. As long as the private key of digital certificate is protected safely, we can take the security advantage of digital certificates (Pfleeger & Pfleeger, 2006).

Digital certificate, however, is slower and harder to create and manage than password. They need to be unique to each other and contain the correct attributes of the end-users. End-users need to fill a list of questions to generate a single certificate. Digital certificates are stored as a digital file, and end-users are in charge of maintaining them on their computers. If the end-users' computer are not secure, their private keys are at risk (Ellison & Schneier, 2000). For a large institution, end-users usually prefer passwords than managing multiple digital certificates.

### 3.2.4 Biometrics

Biometrics deals with identification of individuals based on their biological or behavioural characteristics (Jain et al., 1999). Biometric authentication devices can recognise the following biometrics: fingerprints, hand geometry (shape and size of fingers), retina and iris (i.e. parts of the eye), voice, handwriting, blood vessels in the finger, and face (Pfleeger & Pfleeger, 2006).

Biometrics are less likely to be forged, lost, lent or stolen than passwords. In addition, it can not be forgotten and it is always available. Therefore, it's considered as a more secure authentication solution than username/password authentication mechanism.

Biometrics presents some benefits over traditional password, however, it also presents many issues. Jain et al. (1999); Smith (2001); Schuman (2006); Pfleeger & Pfleeger (2006) list the following issues :

- Biometrics authentication is relatively new when comparing to username/password authentications, and some people find hand geometry and face recognition are inva-

sive. People also have concerns when peering into a laser beam or sticking a finger into a slot.

- Biometric recognition devices are costly. Outfitting every end-user's workstation with a reader can be expensive for a large company with many employees.
- All biometric readers use sampling and establish a threshold for accepting matches. The device has to sample the biometric, measure hundreds of key points, and compare that set of measurements with a template. Variation such as vocal inflection can reduce accuracy.
- Biometrics can become a single point of failure. If recognition failed, the authentication agent (E.g. finger) can not be replaced as easy as a password. Failing biometric authentication is not end-user's fault.
- Although equipment is improving, there are still false readings. We label a "false positive" or "false accept" a reading that is accepted when it should be rejected (that is, the authenticator does not match) and a "false negative" or "false reject" one that rejects when it should accept. Often, reducing a false positive rate increases false negatives, and vice versa.
- The speed at which a recognition must be done limits accuracy. We might ideally like to take several readings and merge the results or evaluate the closest fit. But this is very time consuming and not suitable when the end-users want to start work as soon as possible.
- Forgeries of one's biometrics are possible. Matsumoto et al. (2002) presented artificial fingers that are easily made of cheap and readily available gelatin. They accepted by extremely high rates by particular fingerprint devices with optical or capacitive sensors.

### 3.3 Authentication Models

Authentication model consists of two parties, identity provider (IdP) and service provider (SP). Identity provider manages the identities (e.g. User authentication) of the end-users. Service provider only provides the service (e.g. email services) to the end-users. The emergence of the Internet and distributed system means many computer services became distributed. For example, file storage now can be shared by many end-users in a network. Time-sharing computer also became distributed. Many end-users may share the same operating system for their tasks over computer network. This presents many challenges to authentication models. It must support the massive growth of end-users from all over the Internet in an efficient way. This section gives a detailed background on the evolution of authentication models.

#### 3.3.1 Silo Model

The silo model (or Isolated User Identity Model) is most common identity management system currently deployed on the Internet (Jøsang & Pope, 2005; Jøsang et al., 2007; Vacca et al., 2010). This authentication model combines the identity provider (IdP) and service provider (SP) together in one network domain (Figure 3.1). However, it is not mandatory for them to reside in a single computer system. To access the service providers in a network domain, end-users need to store a copy of their credentials in the identity provider of that domain. And the identity provider of that domain will manages all the authentication processes.

There are two scenarios in silo model. In scenario 'A', identity provider and services provider reside in a single computer system. In scenario 'B', identity provider and service provider reside in separate computer systems, but in the same network domain.

- In scenario 'A', identity provider stores and manages all the credentials locally. If end-users want to access the computer services of that computer system, they must have their credential stored on that computer system. All authentication processes

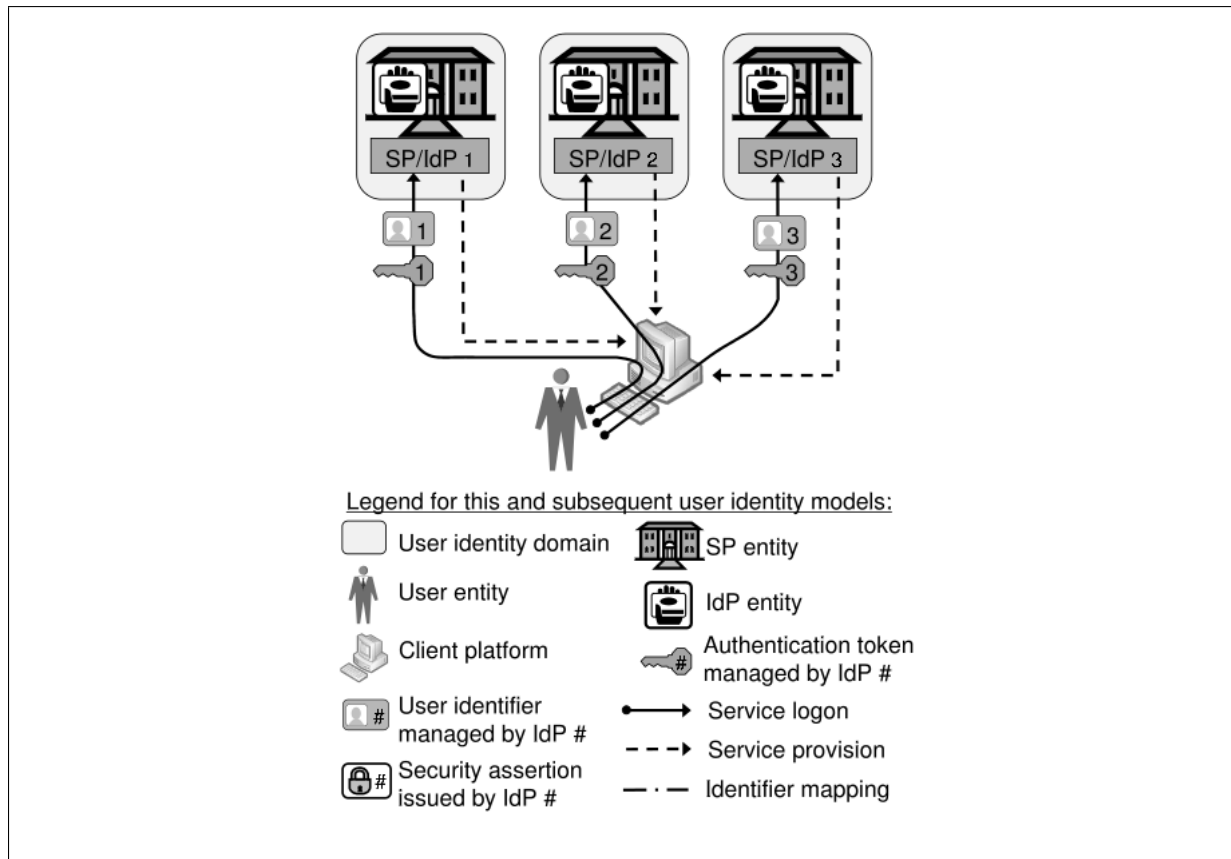


Figure 3.1: Silo Model (Jøsang et al., 2007)

are executed locally on that computer system.

- In scenario 'B', an identity provider stores the credentials of all the end-users in a centralised computer system. The services of a domain rely on the domain's identity provider to provide authentication. All authentication processes are executed over that domain's computer network.

Silo Model is the easiest authentication model to implement and manage. While scenario 'A' only concerns a single computer system, it does not benefit from the central identity provider. However, scenario 'B' benefits greatly from this silo model. Credentials of end-users are stored and managed in a single space instead of multiple systems. For a single domain, end-users only need to remember one set of credentials. It greatly reduces the number of redundant credentials.

It, however, faces challenges when applying to multi-domain environment such as the Internet. The explosive growth in the number of separate online services results in

the Internet becomes a huge multi-domain environment. The approach of silo model may reduce the number of credentials for a single domain, it does not reduce credentials for multiple domains. For each domain and its computer services, end-users need to create and remember sets of credentials. It results in end-users being overloaded with identifiers and credentials that they need to manage (Figure 3.2). End-users are often required to memorise passwords, which unavoidably leads to users forgetting passwords or using the same password combinations for computer services. For example, the latest security breach of online clothing and apparel retailer Zappos.com, highlights the importance of password security. End-users of Zappos.com use the same password for Zappos.com and other services. Zappos.com reset and expired existing passwords for all 24 million customer accounts (Rashid, 2012). In addition, instructions on how to create a secure password were sent to these customers.

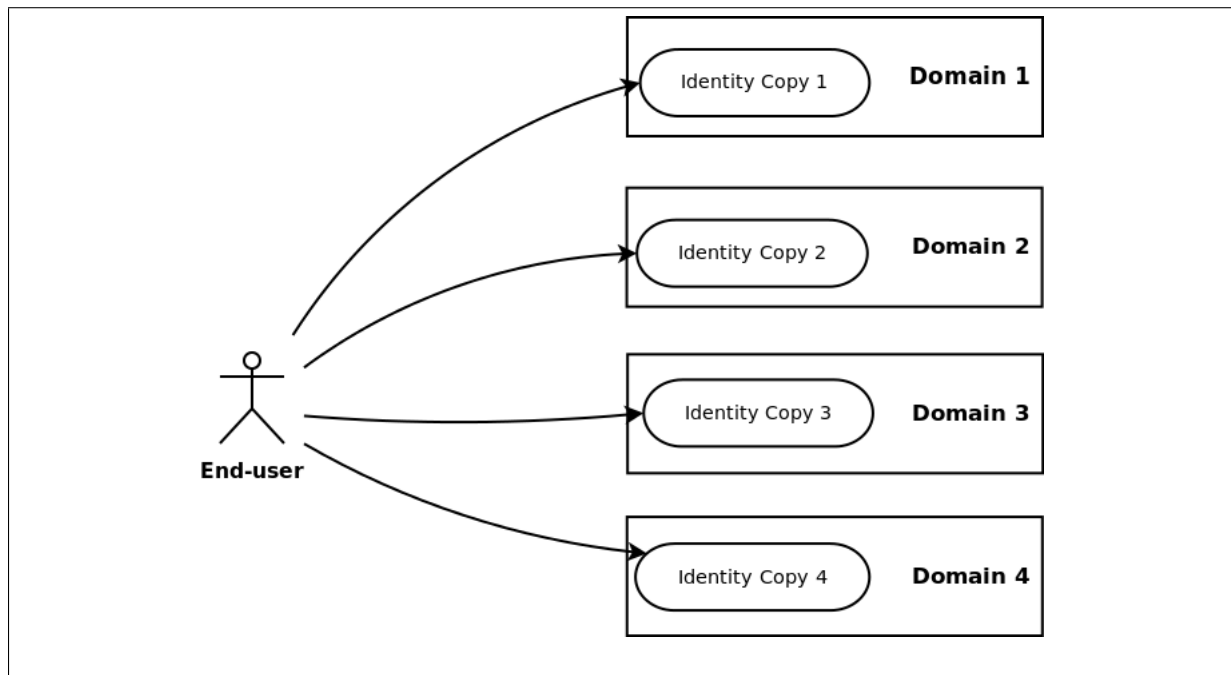


Figure 3.2: An end-user must store credential in every domain

### 3.3.2 Centralised Identity Management (CIM)

To address the problem of redundant credentials in multi-domain environment as well as providing a single authentication session for all the domain accesses, a centralised identity management approach was proposed. It separates the identity providers from



the service providers and merges all the identity providers into a single central identity provider. So that each domain will only contains service providers. The singular identity provider provides identity and authentication for every end-user in the world(Figure 3.3). End-users can have access to all service providers using the same set of identifiers and credentials (Bhargav-Spantzel et al., 2007).

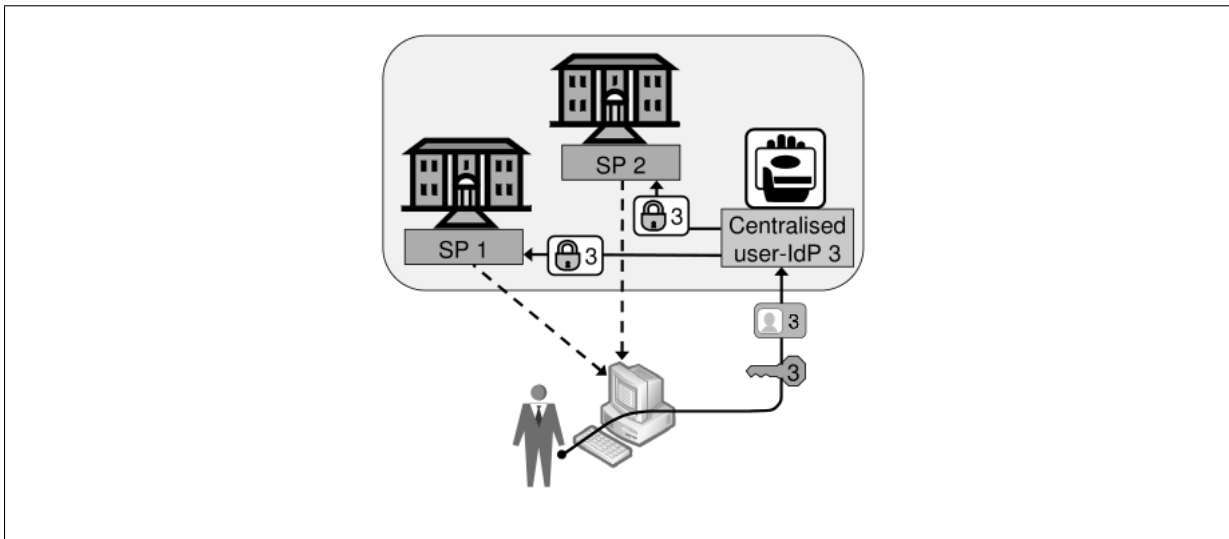


Figure 3.3: Centralised Management Solution (Jøsang et al., 2007)

While centralised identity management allows end-users to have a single identity for every service provider, and it also allow end-users to access multiple service providers with a single authentication session provided by the singular identity provider. However, the main issue of this approach is trust. A singular identity provider means all the credential in the world will be stored in a single space. It is not ideal for enterprises to store their credentials in the same space with the others. It also creates security issues, a centralised identity management system will face a huge security threat from malicious agents. In addition, it suffers from single point of failure. If the identity provider experience technical failures, every service provider in the world will be affected.

### Microsoft Card Space

To give an example, Microsoft's Passport authentication service was introduced in 1999. The idea was that users would get a single, convenient method for identifying themselves across different websites, and thereby stimulate convenient identity transactions (Jøsang

et al., 2007). With Passport, Microsoft will act as the central identity provider and hold end-users' personal information (e.g. credit card numbers) and make it available to all service providers whenever needed.

However, the reaction to Passport was negative. From a privacy perspective, Microsoft will be in charge of all the end-users' personal information in the world. Microsoft will have the significant power to abuse these informations, which a violation of end-users' privacy. In the end, Microsoft acknowledged that "... no single organisation, not even one as big as Microsoft, could act as the sole identity provider for everything on the Internet (Chappell, 2006)." The Passport service was renamed to Windows Live ID in 2006, and currently operates as identity provider for online services owned by Microsoft (e.g. Hotmail).

### 3.3.3 Centralised Single Sign-on

For a single computer domain, there may be many computer service providers, each supports a unique computing task. In order to secure these computer services providers, identity provider must identify legitimate end-users from all the service requests. However, this also increase the number of authentication processes for a single end-user. For each service, end-users need to perform authentication process.

Authenticating to multiple systems is unpopular with end-users. Usually, end-users will reuse the same password to avoid having to remember many different passwords. For example, users become frustrated at having to authenticate to a computer, a network, a mail system, an accounting system, and numerous web sites. When end-users choose weak password or use the same password for every service, it is a security risk for every computer service.

In addition to centralised identity management, CIM also proposed a centralised single sign-on model. Even though due to privacy issues in multi-domain environment, centralised identity management did not came to reality. However, by scaling centralised single sign-on (or single sign-on) model to a single domain, it avoided privacy issues.

Therefore, single sign-on can be used in a single domain environment.

In a single domain, single sign-on addresses the redundant authentication processes by building a mutual trust between computer services providers and the central identity provider. When computer services trust the same identity management system, computer services will trust the authentication processes of the others and grant the access without re-promoting the authentication challenge. An end-user authenticates once, and the system forwards that authenticated identity to all other computer services that require authentication. For example, an authentication session was created when an end-user authenticates to an electronic mail service. With this authentication session, an end-user can access other computer services such as printer service without re-authentication. Therefore, it reduces the number of authentication processes. For example, Kerberos is a single sign on protocol developed and maintained by MIT. Following is a detail review of Kerberos.

### **Kerberos Single Sign-On System**

Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorised users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services. Stallings (2006) listed the following three threats:

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

Kerberos was developed to address this problem (Miller et al., 1988). It enables network applications to securely identify their peers. To achieve this, the client, who requests access to a server, conducts a three-party message exchange to prove its identity to the server (the contacted party) (Kohl et al., 1991). According to Needham & Schroeder (1978), the client proves its identity by presenting to the server a ticket which identifies a principal and establishes a temporary encryption key that may be used to communicate with that principle, and an authenticator which proves that the client is in possession of the temporary encryption key that was assigned to the principal identified by the ticket. The authenticator prevents an intruder from replaying the same ticket to the server in a future session.

The Key Distribution Centre (KDC), which was proposed by Needham & Schroeder (1978), is trusted to hold secret keys known by each client and server on the network (their secret keys are established out-of-band or through an encrypted channel). It also has access to the database that stores end-users' credentials. It authenticates the end-users and creates ticket granting tickets (TGT) for the clients.

Ticket granting tickets (TGT) are issued by a trusted third party KDC. It identifies the client, specifies the session key, lists the start and expiration times, and is encrypted in the key shared by the KDC and the ticket granting server (Denning & Sacco, 1981; Kohl et al., 1991; Venkataramappa, 2002).

Ticket granting server (TGS) is logically distinct from KDC which provides the initial ticket granting ticket service. TGS, however, runs on the same host and has access to the same database of clients and keys used by the KDC (Kohl et al., 1991; El-Hadidi et al., 1997). A client presents its TGT and other request data to the TGS; the TGS verifies the ticket, authenticator and accompanying request, and replies with a ticket for a new server.

Created and maintained by MIT, Kerberos is supported by protocols such as SSH, POP3, SMTP, NFS, DNS and LDAP (Table 3.1). In addition, Microsoft Active Directory and Microsoft service (e.g. file sharing) also supports Kerberos (Samba Team, 2006).

Protocol	Integration Strategy	Specification
Simple Mail Transfer Protocol (SMTP)	SASL+TLS	RFC4754
Post Office Protocol (POP)	SASL+TLS	RFC 5034
Secure Shell (SSH)	GSS-API channel binding to SSH transport	RFC 4462
Network File System (NFS)	GSS-API	RFC 2203
Domain Name System (DNS)	GSS-API for transaction signatures	RFC 3645
Lightweight Directory Access Protocol (LDAP)	SASL+TLS	RFC 4513

Table 3.1: Protocols that Already Support Kerberos (MIT, 2008)

### 3.3.4 Federated Single Sign-on

Centralised single sign-on model has limitation when comes to multiple domain single sign-on. Single sign-on model is based on centralised identity management model. In order for centralised single sign-on to work in multi-domain environment, a central identity provider must be used for all of the service provider in the world. Like centralised identity management model, centralised single sign-on is effected by privacy problems.

Instead of keeping all of the identity providers in a single space, federated single sign-on (FSSO) de-centralises the identity providers. Each domain manages their own identity providers in a way that is similar to silo model. However, unlike silo model, FSSO establishes a federation or a circle of trust (Wason et al., 2003) between all the domains. Mather et al. (2009) defines federation as *the process of managing the trust relationships established beyond the internal network boundaries or administrative domain boundaries among distinct organisations*. Computer domains that in the federation come together to exchange information about their end-users and computer resources and to enable collaborations and transactions (Ahn & Ko, 2007). Then base on that trust, FSSO uses claim based authentication method to allow domains to assert their end-users to the other domains in the federation. This way, an end-user only needs to authenticate to the identity provider of one domain and access services from other domains based on the trust (Figure 3.4).

Unlike centralised single sign on technologies, federated single sign-on model uses identity assertion model, which means every domain will assert its end-users to the other

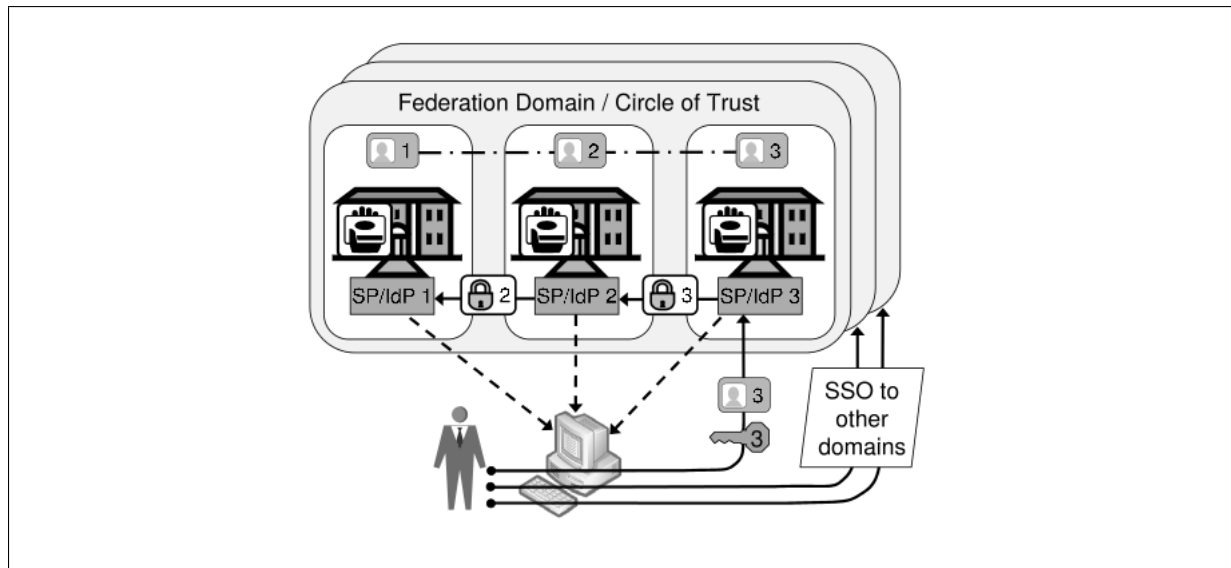


Figure 3.4: Federated Single Sign-On (Jøsang et al., 2007)

domains (Patterson et al., 2004). Assuming that a federation has been established between domain Alpha and Beta and a set of policies have been established. The policy indicates that end-users from domain Alpha can access computer services in domain Beta. According to Ying (2010); Takaaki et al. (2011); Sato & Nishimura (2011); SWITCHaai (2012); Google Inc. (2012), federated single sign-on consists of the following process (figure 3.5):

1. An end-user requests access to a service application in domain Beta.
2. The service provider in domain Beta asks the domain which the end-user belongs to.
3. The end-user selects the domain (in this case, domain Alpha).
4. Service provider in domain re-directs the end-user to domain Alpha's identity provider.
5. The end-user authenticates to the identity provider in domain Alpha.
6. If the authentication was successful, the identity provider in domain Alpha will create an identity assertion for the end-user. The assertion indicates that the end-user belongs to domain Alpha.
7. The end-user presents the identity assertion to the service provider in domain Beta.

8. Service provider accepts the end-user's request to its computer services.

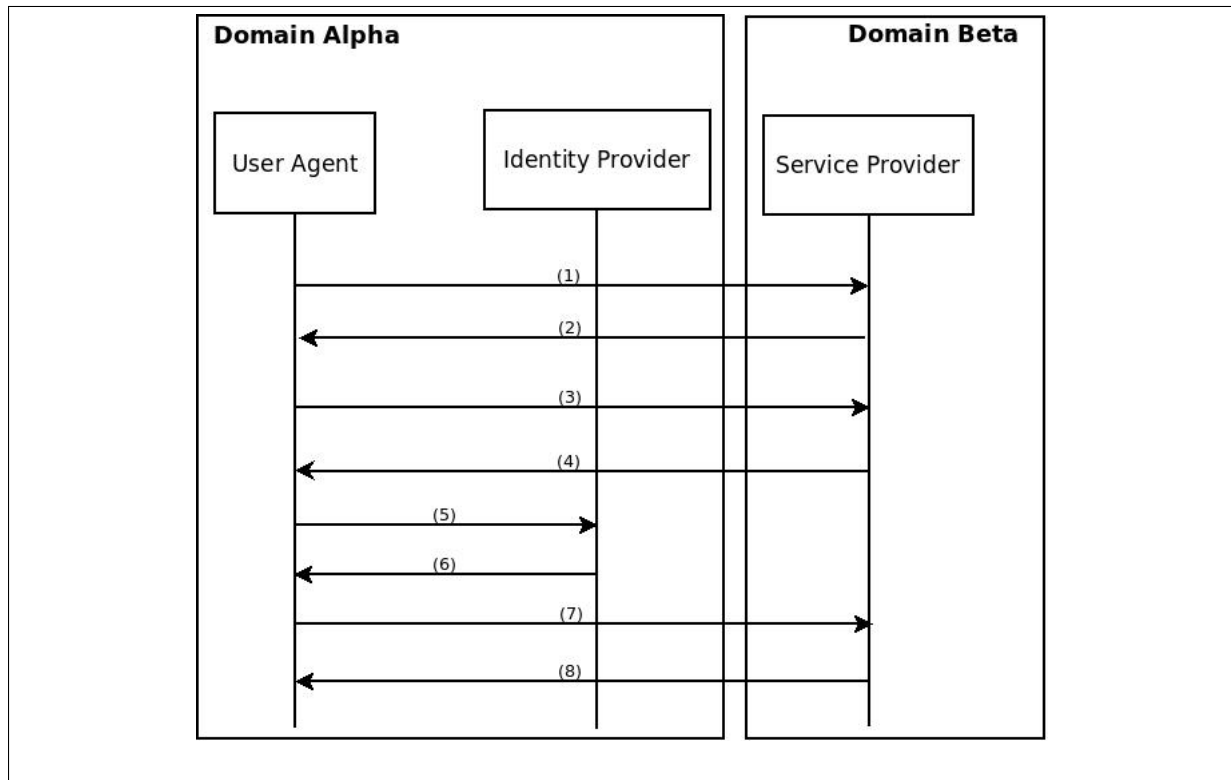


Figure 3.5: Federated Single Sign-On Process

Federated single sign-on was initially developed for web-based applications (Don & Smith, 2008; Paul & Madsen, 2004). The presented federated single sign-on processes is used by web-based applications. Followings are approaches of federated single sign-on for web applications:

- Security Assertion Markup Language (SAML)
- WS-Federation
- Liberty Identity Federation Framework (ID-FF)
- Shibboleth Project

Although federated single sign-on was initially developed to support web-based applications, Howlett (2011); Gridwise Tech (2010) Followings are approaches of federated single sign-on for non-web based applications:

- Project Moonshot

- Kerberos Secure Sharing

Following are approaches of federated single sign-on for both:

- SASL-SAML
- SAML-AAI/Kerberos

## 3.4 Summary

Authentication is one of the core functionalities of identity management. It is the process of verifying claims about holding specific identities. There are many authentication mechanisms: username/password, challenge and response, digital certificate and biometrics were developed. All of these authentication mechanisms address a set of issues, however, they also present new issues. There is not a single perfect solution, end-users and organisations need to balance their costs, secure requirements, so they can choose the best solution for themselves. Currently, username/password is the most widely used authentication mechanism.

This chapter also presents authentication models such as Silo model, Centralised identity management model and Federated single sign-on model. The research found that the silo model is widely used in a single domain environment. In a multi-domain environment, Silo model presents issues such as redundant credentials and redundant authentication processes. Centralised identity management aims to address the redundant credential issue by providing a global identity provider. However, Centralised identity management causes huge trust issues, and it did not become reality. Federated single sign-on aims to address the redundant credential and redundant authentication processes by building trust between domains. With that trust, domains can assert their end-users to the other domains instead of providing credentials. This approach has the ability to scale up to multiple domains, it also allows an end-user can access diverse computer services (in multiple domains) with one authentications session. Chapter 4 will give an extensive literature review of the state of the art researches on federated single sign-on.



# Chapter 4

## Federated Single Sign-On (FSSO) Systems

### 4.1 Introduction

Before introducing federated single sign-on (FSSO), it is important to introduce some pre-requirement of FSSO first. Federated single sign-on requires the establishment of a circle of trust (Sullivan, 2005) or a federation (Mather et al., 2009). Sullivan (2005) describes a circle of trust as *“usually composed of a group of service providers who share linked identities and who have pertinent business agreements in place regarding how to do business and interact with identity providers.”* Mather et al. (2009) describes a federation as *“the process of managing the trust relationships established beyond the internal network boundaries or administrative domain boundaries among distinct organisations.”* Ahn & Ko (2007) states *“network domains in the federation come together to exchange information about their end-users and computer resources and to enable collaborations and transactions.”* Based on these descriptions, this research found that the term “circle of trust” and the term “federation” can be used interchangeably.

The cycle of trust is expressed in the form of security policies. These policies ensure the end-users have the correct level of access to network domains’ computer resources.

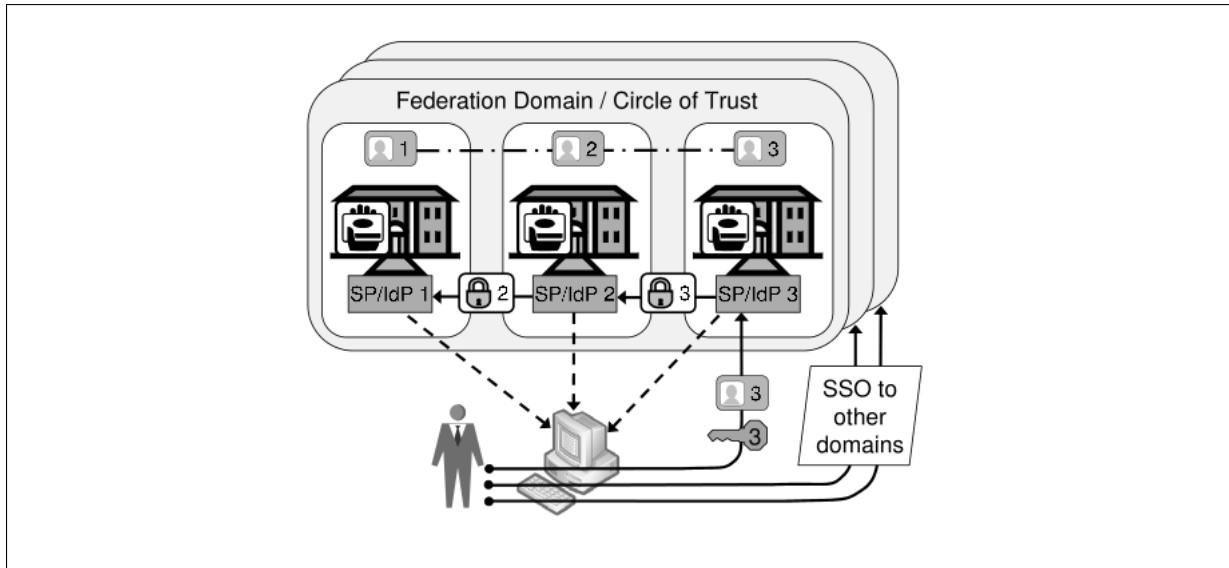


Figure 4.1: Federated Single Sign-On (Jøsang et al., 2007)

Organization for the Advancement of Structured Information Standards gave an sample policy in Godik & Moses (2003). The policy shows a corporation named Medi Corp (medico.com) has an access control policy that states, in English: Any end-user with an e-mail name in the “medico.com” namespace is allowed to perform any action on any resources. The policies may be changed to machines readable language, however, policy management is outside the scope of this research.

In a federation, a federated single sign-on (FSSO) model allows the use of the same end-user’s personal identification information (PII) across multiple organisations (Suriadi et al., 2009). Once an end-user has been authenticated by an identity provider within a federation, that identity provider can provide identity assertion for that end-user. The identity assertion indicates that the end-user is who he/she claims to be and he/she belongs to one of the part within the federation. And with the identity assertion, that end-user can access service providers within the federation without authentication again (single sign-on within a federation) (Patterson et al., 2004). This way, an end-user only needs to authenticate once to their home domain and be-able to access computer services providers within a federation (Figure 4.1).

As presented in section 3.3.4, federated single sign on model also includes two components, identity provider and service provider. Identity provider (IdP) manages the

private identity of the end-users. Service provider (SP) focus on providing the end-users with computer services. Doesn't matter which authentication model is in place, both parties will always in charge of the same task. The difference between federated single sign-on (FSSO) and the other is that the identity providers are de-centralised to individual computer domains (Table 4.1). Computer domains implement and manage their own identity providers. Then a circle of trust (or federation) was established between the domains. In authority-centric federated single sign-on, identity providers provide identity assertions for authenticated end-users.

	<b>Silo Model/Isolated User Identity Model</b>	<b>Centralised Identity Management Model</b>	<b>Federated Single Sign-On Model</b>
Ownership of identity provider	Every domain manages its own identity providers.	An single identity provider is shared between domains.	Every domain manages its own identity providers.
Ownership and management of identities	Identities are owned and managed by their associate domains' identity providers.	Identities are owned by their associate domains but managed by a single identity provider.	Identities are owned and managed by their associate domains' identity providers.
Location of identities	One domain's identities are isolated from the other domains' identities.	Identities reside in a single space.	One domain's identities are isolated from the other domains' identities, however, identities' attributes can be shared between domains.

Table 4.1: Difference Between Silo Model, Centralised Identity Management Model and Federated Single Sign-On Model

Beside the traditional "authority-centric" federated single sign-on system that was described, there is also a user-centric federated single sign-on system (Table 4.2). The user-centric federated single sign-on (UFed SSO) aims to address the privacy issue that reside in the federated single sign-on system (Suriadi et al., 2009). Federated single sign on concentrates end-users' privacy identification information into identity providers. Each identity provider may track some of the end-users' activities. UFed SSO adopts the user-centric identity management (UCIM) and designs from the end-user's perspective (Jøsang & Pope, 2005). In this system, end-users have an effective control of the use and management of their private identification information. According to Suriadi et al. (2009), UCIM mechanism provides end-users with means (e.g. storage) to manage their private identification information. The user-centric nature of this systems attracted many

web services providers such as Facebook, Google Inc and Yahoo.com. OpenID, which is an implementation of user-centric federated single sign-on model, is widely used by these enterprises. Any customer of Google Inc.'s email service, gmail, can use the same credential for Youtube.com (an online video sharing service) and Google Apps (Google Inc.'s web-base document editor suite). End-user sign into gmail.com and can use the other services without re-authentication.

	<b>Authority-centric FSSO</b>	<b>User-centric FSSO</b>
Identity claim	The end-user was verified as who he/she claim to be (strong claim).	The end-user has an identity (weak claim).
Management of identity information	Identity information is managed by the identity providers.	Identity information is managed by the end-users.
Ownership of identity information	Identity information is owned by identity providers.	Identity information is owned by end-users.

Table 4.2: Authority-centric and User-centric Federated Single Sign-On

Due to the complexity of identity management, not all of the end-users have the knowledge to manage their identities. Also, end-users may not provide enough security measures to protect their identity information. Existing solution such as OpenID requires end-users to register in a services vendor first, therefore, end-users are still effected by privacy issues. In addition, without corroboration by a trusted third party, service providers are not obliged to trust any claim of any end-user (Maler & Reed, 2008). Therefore, “authority-centric” federated single sign-on is still favourite by business enterprises, governments and education faculties such as JANET (2012) and HEAnet (2012b).

The focus of this research resides in the “authority-centric” federated single sign-on. Research in user-centric federated single sign-on is out of the scope of this research. Section 4.2 presents the state of the art federated single sign-on standards in web space. Section 4.3 presents the state of the art federated single sign-on solutions in non-web space. Section 4.4 presents the state of the art federated single sign-on solutions in both web-based and non-web based space. Section 4.5 presents the summary of this chapter.

## 4.2 Federated Single Sign-On Solutions in Web Space

According to Don & Smith (2008), federation technologies first appeared with the aim of offering web-based single sign-on across organisational domains. Paul & Madsen (2004) also emphasised the relationship between web services and federated single sign-on technologies. According to Paul & Madsen (2004), federated identity and web services both complement and are dependent on each other. Federated identity standards such as Security Assertion Markup Language (SAML) and ID-Web Service Framework take advantage of the web services standards stack. In this sense, federated identity standard was seen as built on top of web services.

Federated single sign-on standard such as OASIS' (Organisation for the Advancement of Structured Information Standards) Security Assertion Markup Language (SAML), Liberty Identity Federation Framework (ID-FF) and Shibboleth Project were designed for web services. Both standards are described in section 4.2.1, section 4.2.2 and section 4.2.4. Based on these standards, many vendors implemented their own version of federated single sign-on solutions such as Ping Federated (Ping Identity, 2012). In August 2005, vendors such as Sun, Oracle and Novell were amongst eight who secured the Liberty Alliance's seal of approval on SAML 2.0 interoperability testing (Don & Smith, 2008). In addition, enterprises such as IBM and Government Agencies such as the US General Services Administration and the Department of Defense (DOD) have joint Liberty Alliance in support of federated single sign-on standards (William & Knight, 2004; ScienceDirect, 2003).

This section aims to present the state of the art federated single sign-on standards—SAML and ID-WSF. Because SAML had evolved from V1.x to V2.0. Therefore, there will be two sections dedicated to SAML. In addition, this section also presents the WS-Federated Authentication Module. It's a federated authentication model designed and implemented by Microsoft®.

### 4.2.1 Security Assertion Markup Language (SAML) V1.0

According to Maler et al. (2003), SAML is an XML-based framework for exchanging security information (e.g. end-user's authentication, entitlement and attribute information). This security information is expressed in the form of assertions about subjects, where a subject is an entity (either human or computer) that has an identity in some security domain. For example, end-users identified by their email addresses such as joe-doe@example.com. It was developed by the Security Services Technical Committee of the Organisation for the Advancement of Structured Information Standards (OASIS). It allows business entities to make assertions regarding the identity, attributes, and entitlements of a subject (an entity that is often a human user) to other entities, such as a partner company or another enterprise application.

SAML relies on identity assertions for cross domain authentication. Assertions can convey information about authentication acts that were previously performed by subjects, attributes of subjects, and authorisation decisions about whether subjects are allowed to access certain resources. A single assertion might contain several different internal statements about authentication, authorisation, and attributes (Maler et al., 2003).

Assertions are issued by SAML authorities, namely, authentication authorities, attribute authorities, and policy decision points. SAML defines a protocol by which clients can request assertions from SAML authorities and get a response from them. This protocol, consisting of XML-based request and response message formats, can be bound to many different underlying communications and transport protocols; SAML currently defines one binding, to SOAP over HTTP.

SAML authorities can use various sources of information, such as external policy stores and assertions that were received as input in requests, in creating their responses. Thus, clients always consume assertions, SAML authorities can produce and consume assertions.

The following model (Figure 4.2) is conceptual only; for example, it does not account for real-world information flow or the possibility of combining of authorities into

a single system.

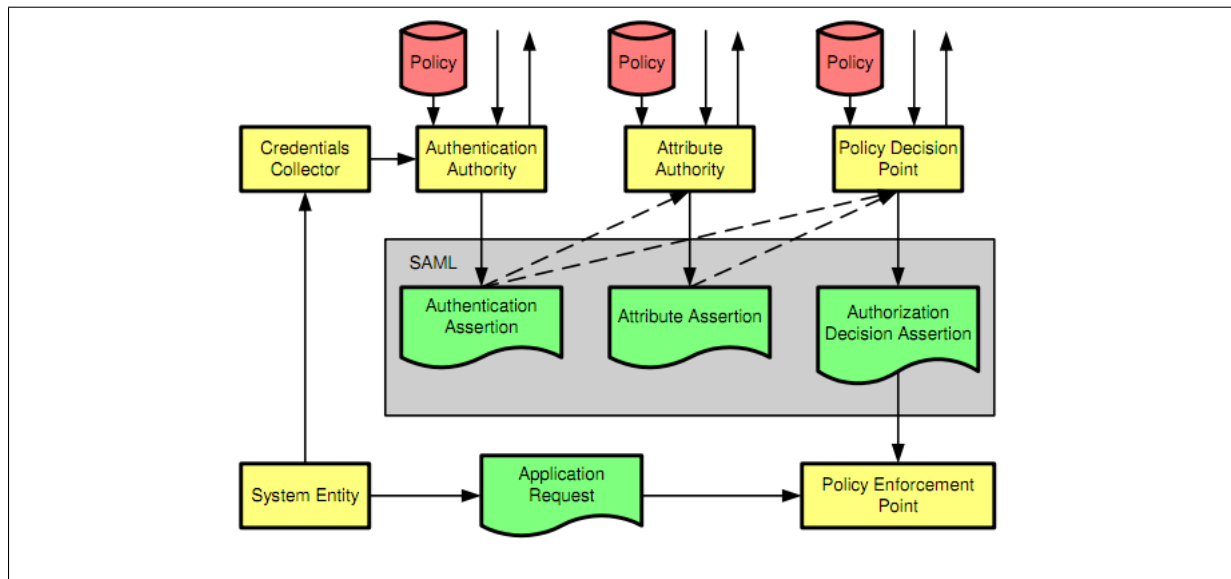


Figure 4.2: The SAML Domain Model

One major design goal for SAML is Single Sign-On (SSO), the ability of a user to authenticate in one domain and use resources in other domains without re-authenticating. However, SAML can be used in various configurations to support additional scenarios as well. Several profiles of SAML have been defined that support different styles of SSO, as well as the securing of SOAP payloads.

The characteristics of SAML includes:

1. Platform neutrality. SAML abstracts the security framework away from platform architectures and particular vendor implementations. Making security more independent of application logic is an important tenet of Service-Oriented Architecture;
2. Loose coupling of directories. SAML does not require user information to be maintained and synchronised between directories;
3. Improved online experience for end users. SAML enables single sign-on by allowing users to authenticate at an identity provider and then access service providers without additional authentication;
4. Reduced administrative costs for service providers. Using SAML to “reuse” a single act of authentication (such as logging in with a username and password) multiple

times across multiple services can reduce the cost of maintaining account information;

5. Risk transference. SAML can act to push responsibility for proper management of identities to the identity provider, which is more often compatible with its business model than that of a service provider;

### **Shibboleth® Project**

The Shibboleth® System software package for web single sign-on, it can be used across or within organisational boundaries. It's based on standards, and it's open source. It allows sites to make informed authorisation decisions for individual access of protected online resources in a privacy-preserving manner.

According to Internet2 (2012), the Shibboleth software uses Security Assertion Markup Language (SAML) to provide a federated single sign-on and attribute exchange framework. Shibboleth also provides extended privacy functionality allowing the browser user and their home site to control the attributes released to each application. It simplifies management of identity and permissions for organisations supporting users and applications. Shibboleth is developed in an open and participatory environment, is freely available, and is released under the Apache Software License.

### **4.2.2 Liberty Identity Federation Framework (ID-FF)**

The Liberty Alliance Project was formed in September 2001 to develop open standards for federated network identity management and identity-based services (Sun Microsystems, 2004). Its goals are to ensure interoperability, support privacy, and promote adoption of its specifications, guidelines, and best practices.

Liberty Identity Federation Framework (ID-FF) is built on top of existing XML-based standards and enables identity federation and management through single sign-on, user account linking, and simple session management (Shim et al., 2005). Various Web-



based service providers form a federated network with an identity provider—an entity responsible for creating, maintaining, and authenticating all user identities—that enables users to securely operate among network members. Users need only sign on once with any member to access Web sites in the circle of trust. Improve upon SAML v1.x, ID-FF includes the following functionalities that are not in SAML v1.x:

- global sign-outs,
- attribute sharing between providers.

The key objectives of the Liberty Alliance are to:

- Enable consumers to protect the privacy and security of their network identity information,
- Enable businesses to maintain and manage their customer relationships without third-party participation,
- Provide an open single sign-on standard that includes decentralised authentication and authorisation from multiple providers,
- Create a network identity infrastructure that supports all current and emerging network access device.

ID-FF links domains' service providers and identity providers' accounts through pseudonyms—arbitrarily assigned names that have meaning only in the context of a given exchange. This eliminates the need for global identification and prevents collisions between different service providers. The framework also guarantees anonymity by allowing service providers to request certain attributes without knowing the users identity (Shim et al., 2005).

### 4.2.3 WS-Federated Authentication Module

WS-Federation is a federated identity standard being jointly developed by Microsoft, IBM, RSA Security, VeriSign, and BEA. It relies on a security token service to broker

trust of identities, attributes, and authentication between participating Web services. According to Microsoft (2012b), WS Federation Authentication Module (WS-FAM) is an HTTP module that adds federated authentication to a ASP.NET application. Federated authentication lets authentication logic be handled by the Security Token Service (STS) and allow organisations to focus on writing business logic.

Organisations can configure the WS-FAM to specify the STS to which non-authenticated requests should be redirected. WIF lets you authenticate a user in two ways:

1. Using the Federated Passive Sign In control. When an unauthenticated user tries to access a protected resource, they are redirected to a logon page in the application. This logon page has the Federated Passive Sign In control embedded in it and the control is configured with issuer (STS) information. If you do not require application-wide protection and instead require a logon page in the application and you expect the users to click the control (for example, in a financial institution online site with a login page), then this is the right approach;
2. Passive redirect: When an unauthenticated user tries to access a protected resource, and you want to simply redirect them to an STS without requiring a login page, then this is the right approach. The STS verifies the user's identity, and issues a security token that contains the appropriate claims for that user. This option requires the WS-FAM to be added in the HTTP Modules pipeline. For more information, see *Establishing Trust from an ASP.NET Relying Party Application to an STS using FedUtil*;

According to McRae (2009), WS-Federation v1.2 has been approved by OASIS to become a standard of web-based federated single sign-on. Since Microsoft has integrated WS-FAM into their software, WS-FAM will be used by network domains that use Microsoft products.

#### 4.2.4 Security Assertion Markup Language (SAML) V2.0

Recognising the value of a single standard for federated SSO, the Alliance submitted ID-FF V1.2 back into the OASIS Security Services Technical Committee (OASIS) as input for SAML V2.0. Mather et al. (2009); Jason & Goode (2012) state that SAML 1.0, Shibboleth Project and Liberty Alliance combined their work in enhancing SAML 1.0 to create SAML 2.0 (Figure 4.3). SAML 2.0 is the culmination of work stemming from the Organisation for the Advancement of Structured Information Standards (OASIS) (OASIS, 2011), the Liberty Alliance Project, and the Shibboleth Project.

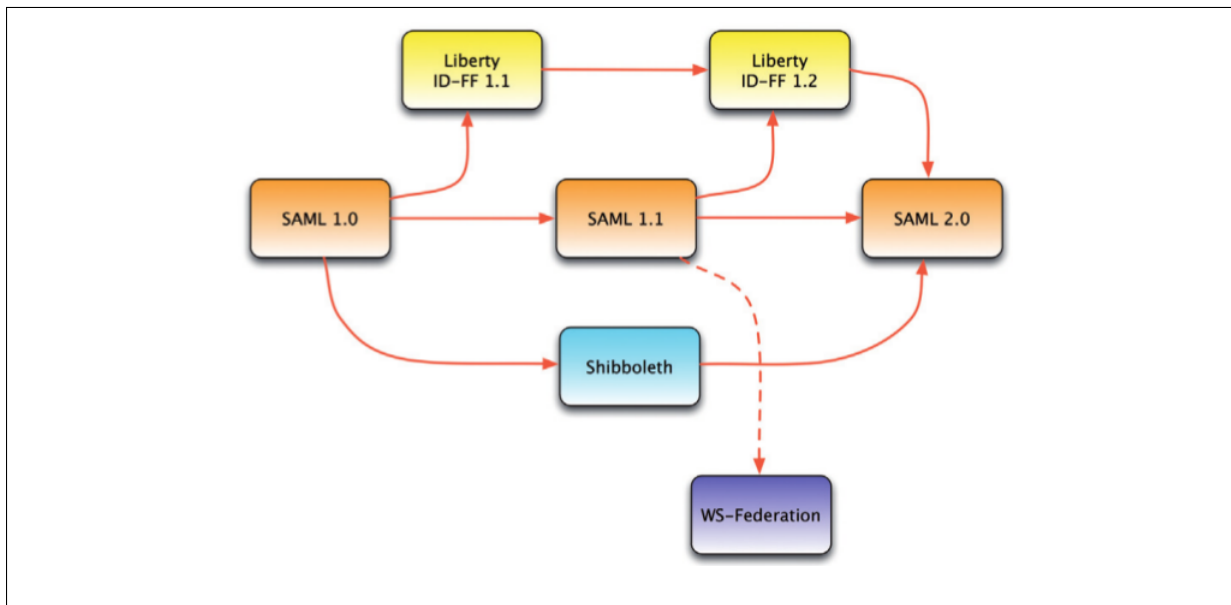


Figure 4.3: SAML 2.0 incorporates three existing federation technologies and standards – SAML 1.1, ID-FF 1.2 and Shibboleth (Jason & Goode, 2012).

Because of the combination, SAML 2.0 emerged as the standard for federated single sign on for web-based applications. Ragouzis et al. (2008) states in the *SAML 2.0 Executive Overview* that SAML has been broadly implemented by all major *web access management vendors*. SAML support also appears in major application server products and is commonly found among *web services management and security vendors*.

Based on the review of SAML1.0, ID-FF and Shibbloth and WS-FAM, it is clear that SAML 2.0 and WS-FAM are the predominant standards for web-based federated single sign-on. Network domains will continue to use web-based federated solutions that are based on these standards. WS-FAM are built into the Microsoft products, therefore,

network domains that use Microsoft products will most likely use WS-FAM. On the other hand, SAML is contributed by OASIS, Liberty Alliance and Shibboleth. SAML based solution such as Shibboleth is open source and free of charge. It is clear that SAML is and will be the de-facto standard for federated single sign-on.

### 4.3 Federated Single Sign-On Solutions Beyond Web Space

Although, federated single sign-on was initially aimed for web space, however, Papez (2009); Wierenga & Lear (2010); Howlett (2011) had emphasised the need of federated single sign-on in beyond web space. Because the aims of standards (e.g. SAML v2.0) are web-based federated single sign-on, the technologies that emerged aim for web-based applications. There is a lack of solution to take this federated identity management to applications beyond the web such as e-mail, instant messaging, file sharing, or remote computing (Painless Security, 2010). In addition, interoperability of web-based and non-web based federated single sign-on still remain as a challenging topic.

State of the art projects such as Moonshot, SASL-SAML and SAML-AAI/Kerberos aim to design federated single sign-on solutions based on the existing Security Assertion Markup Language (SAML) standard. Although SAML was designed for web-based federated single sign-on, however, the platform independent nature of XML indicates that it's plausible to apply its technology to non-web space. Project such as Kerberos Secure Sharing (KSS) aims to explore the Kerberos protocol for cross domain single sign-on. This section analyses these technologies and presents a review of the state of the art technologies.

#### 4.3.1 Project Moonshot

According to Howlett (2011), project Moonshot is a federated single sign-on project initiated by JANET(UK). It aims to address a number of issues with SAML-based federation.

These issues include: the difficulty of using a federated identity with client that are not HTTP user agents; the ‘Identity Provider Discovery’ and ‘Multiple Affiliation’ problem; and inter-federation issue. It extends SAML based federated single sign on to non-web based applications. It proposes to combine two key security technologies: the Extensible Authentication Protocol (EAP) and the Generic Security Services Application Programming Interface (GSS-API) to create a solution for federated authentication. EAP and RADIUS infrastructure provide federation and access to the mechanisms people use to authenticate. GSS-API provides integration with application protocols and applications.

Howlett (2010) described the architecture is to provide a single technical approach for addressing the non-web based single sign on and provide scalable trust. In addition, Howlett states that project moonshot may provide other novel benefits: avoiding the use of keys or key names in SAML metadata, it greatly deduces the issues associated with key management and distribution, by keying from a negotiated GSS context; The ability to use any kind of SAML metadata distribution mechanism, trusted or not; and entities do not need to share a common credential technology. Project Moonshot contains four core components:

- A RADIUS attribute for SAML message. It encapsulates SAML constructs (principally SAML Request and Response protocol elements) in RADIUS attributes between a AAA client and server;
- Binding AAA with SAML 2.0. It specifies the use of RADIUS protocol as a transport;
- A GSS-API Mechanism for the Extensible Authentication Protocol (EAP). It allows the use of any EAP method within GSS-API. This specification takes advantage of the EAP architecture’s concept of a pass-through authenticator, allowing a GSS initiator (the EAP peer) to authenticate to a GSS acceptor (the EAP pass-through authenticator) by reference to a AAA/EAP server that can authenticate the initiator’s EAP credential;
- Name Attributes for the GSS-API EAP mechanism. It is a new GSS-API mechanism

that makes use of the Extensible Authentication Protocol, allowing the use of any EAP method within GSS-API;

- The SAML v2.0 EAP GSS SSO Profile. It is a new SAML SSO profile that is intended to be applied to application protocols supporting GSS or SASL;

Project moonshot's model (Figure 4.4) details the life-cycle of a federated single sign on transaction for non-web based applications. In addition to the transaction process, it also shows that the user agent, identity provider and service provider are modified to support the transaction.

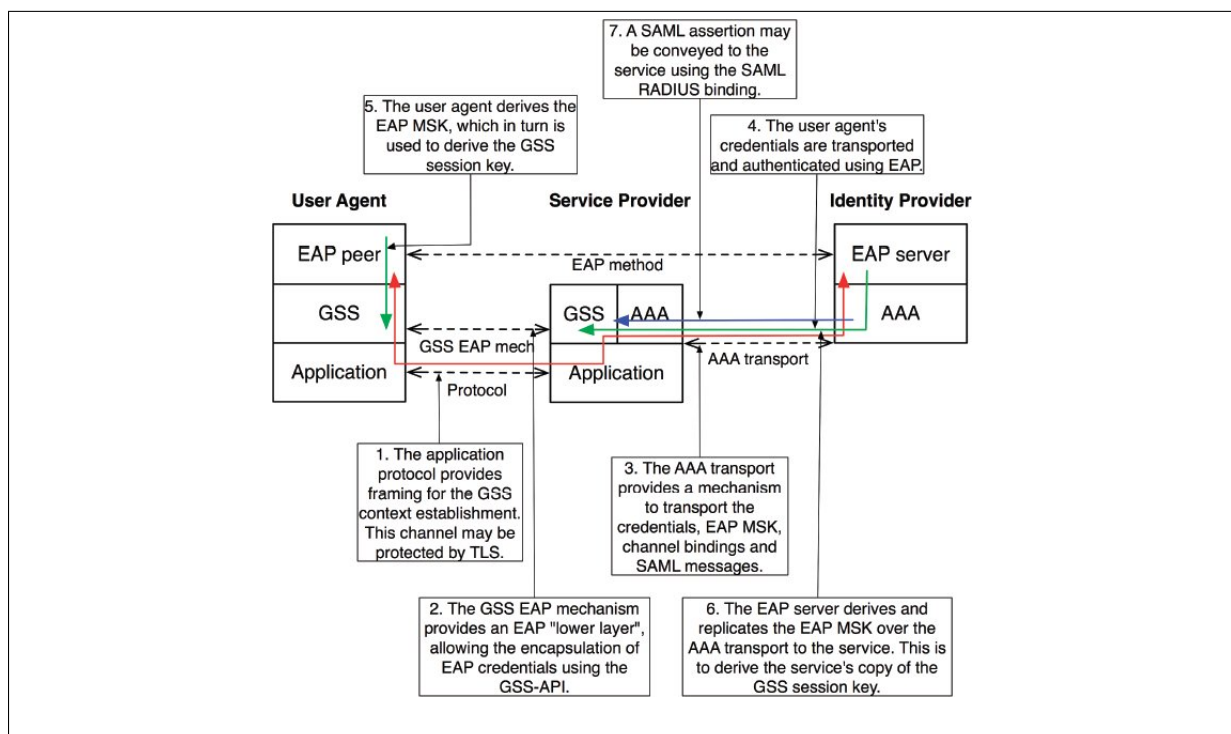


Figure 4.4: Project Moonshot Model (Howlett, 2011)

According to Painless Security (2010), Moonshot makes several important assumptions about the environment of the client:

- Clients have EAP supplicant software installed and configured;
- Clients have configured their supplicant with some information about the sets of identities they use;
- Clients may be required to take some steps to enable support for Moonshot in their applications;

It is clear that Moonshot suites organisations that have the ability to configure the machines of all their affiliates. Moonshot also suites experienced end-users who are able to follow directions and who see sufficient benefit in federated identity management to do so. It, however, is not well-suited to the needs of users who use machines that they do not control or that have not been configured to meet the needs of their organisation.

In addition to applicability, Moonshot aims to build a separate federated single sign on system for non-web based applications. It does not support web-based federated single sign-on. Therefore, if an organisation implements a federated single sign-on technology for web-based applications, there is no guarantee that both technology will interoperate. Project Moonshot also do not leverage any existing authentication infrastructures, if network domains provide their own authentication systems (e.g. desktop authentication using directory services), end-users will need to authenticate with the existing authentication system (to access desktop), Project Moonshot (to access non-web based services/applications) and SAML based federated single sign-on system (to access web-based services/applications). This is not true single sign-on.

### 4.3.2 Kerberos Secure Sharing

Kerberos Secure Sharing (Gridwise Tech, 2010) is an approach that enables federated single sign-on for non-web based applications only (Figure 4.5). It proposes an architecture that implements federated security using LDAP, Kerberos and GridwiseTech's AdHoc. LDAP manages user authorisation to resources. Kerberos manages user authentication using SSO. GridwiseTechs AdHoc responsible for managing access policies to resources. AdHoc is divided into two distinct models. AdHoc web application provides a graphic web interface where users and administrators manage access to their resources. AdHoc policy manager is responsible for modifying the user access policies of the underlying resources.

Because Kerberos is a well established protocol, it faces less deployment issue than project Moonshot. End-users can log in their desktop with Kerberos authentication and access computer services/applications (which also support Kerberos) without re-

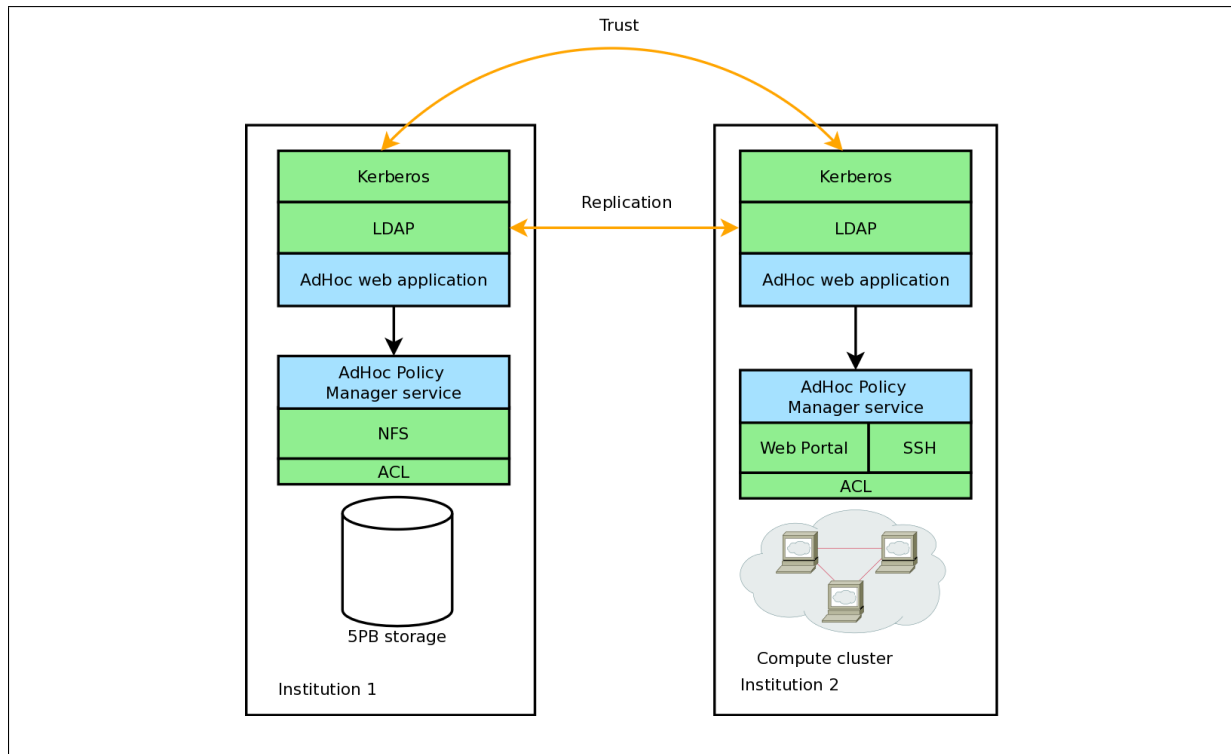


Figure 4.5: Kerberos Secure Sharing (Gridwise Tech, 2010)

authentication. In addition, Kerberos has been supported by web-based identity provider, so that end-users can use Kerberos to authentication to web-based services/applications as well. This provides a true single sign-on experience.

Kerberos Secure Sharing, however, needs to expose the Kerberos key distribution centre (KDC) and Kerberos ticket granting service (TGS) to the other domains. Since these components have direct access to end-user databases, this approach is suitable if domains are willing to expose their KDC and TGS, otherwise, this approach will fail. If a set of network domains have the same owner, they can expose their KDCs to each other, they can use this approach for cross domain single sign-on. If a set of network domains have separate owners, and they are unwilling to expose their KDCs to each other, this model will fail.

While, network domains are reluctant to extend their Kerberos system beyond their domains. It is possible to provide a middle-ware system that link the Kerberos system (of each domain's) together to facilitate federated single sign-on. Since these middle-ware systems do not have direct communication to end-user databases, network domains are more likely to accept them.



Assuming network domains support Kerberos single sign-on system. The middle-ware systems can authenticate end-users with Kerberos single sign-on system and request Kerberos tickets in the other domains for end-users. The Kerberos tickets allow the end-users to access non-web based services/applications without re-authentication.

## 4.4 Federated Single Sign-On Approaches in Both Space

This section reviews the existing approaches that aim to bring federated single sign-on to both web-based and non-web based space. Project SASL-SAML aims to allow end-users' client applications to accept SAML token that transferred through HTTP protocol. SAML-AAI/Kerberos aim to use browser based federated single sign-on to acquire Kerberos token for non-web based applications. All these solutions support the interoperability between web-base and non-web based federated single sign-on.

### 4.4.1 SASL-SAML

SASL-SAML (Wierenga & Lear, 2010) is another approach that extends the support of SAML to non-web based applications. Unlike project Mooshot, this approach does not set up a separate federated single sign-on system. It specifies a SASL mechanism and a GSS-API mechanism for SAML 2.0 that allows the integration of existing SAML identity providers with applications using SASL and GSS-API. Therefore, this approach can be applied to any domain that has a pre-existing web-based federated single sign-on solution.

Simple Authentication and Security Layer (SASL) is a generalised mechanism for identifying and authenticating a user and for optionally negotiating a security layer for subsequent protocol interactions. SASL is used by application protocols like IMAP, POP and XMPP. The effect is to make modular authentication, so that newer authentication mechanisms can be added as needed. The Generic Security Service Application Program Interface (GSS-API) provides a framework for applications to support multiple authenti-

cation mechanisms through a unified interface.

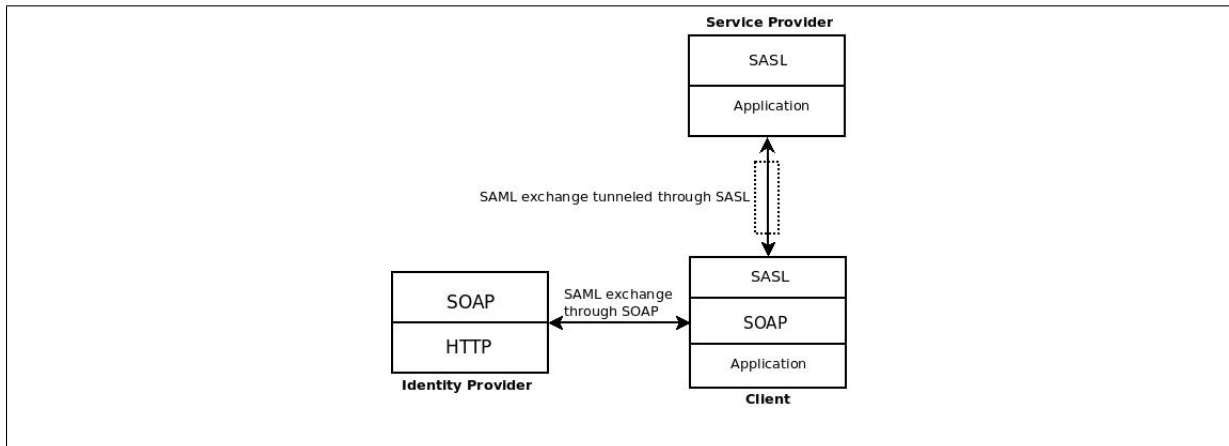


Figure 4.6: SAML-SASL Model

The proposed mechanism is to allow the interworking between SASL and SAML in order to assert identity and other attributes to service providers. As such, while servers (as service providers) will advertise SASL mechanisms (including SAML), clients will select the SASL-SAML mechanism as their SASL mechanism of choice. SASL-SAML relies on the exchange of SAML through SASL channel to achieve federated single sign on for non-web based applications (Figure 4.6).

Similar to Project Moonshot, SASL-SAML does not take advantage of existing authentication infrastructure, and it needs to modify existing computer services/applications. This approach faces the same deployment issue, if applications providers and network domains are unwilling to modify their existing services/applications, this model will fail. In addition, it doesn't deliver a true single sign-on experience.

#### 4.4.2 SAML-AAI/Kerberos

Papez (2009) introduces a hybrid web applications that integrate web-based federated single sign-on with back-end Kerberos protected services (figure 4.7). In this model, an end-user authenticates to a web-based identity provider first. The identity provider creates an identity assertion for the end-user, this assertion will be sent to the service provider in another domain. Assuming that both domains have established a federation. The SAML-AAI/Kerberos model allows the service provider can request Kerberos ticket

granting ticket (TGT) and service ticket (ST) on behalf of the end-user in the other domain. The end-users can use the ST to access computer services in the other domain through web-based user client.

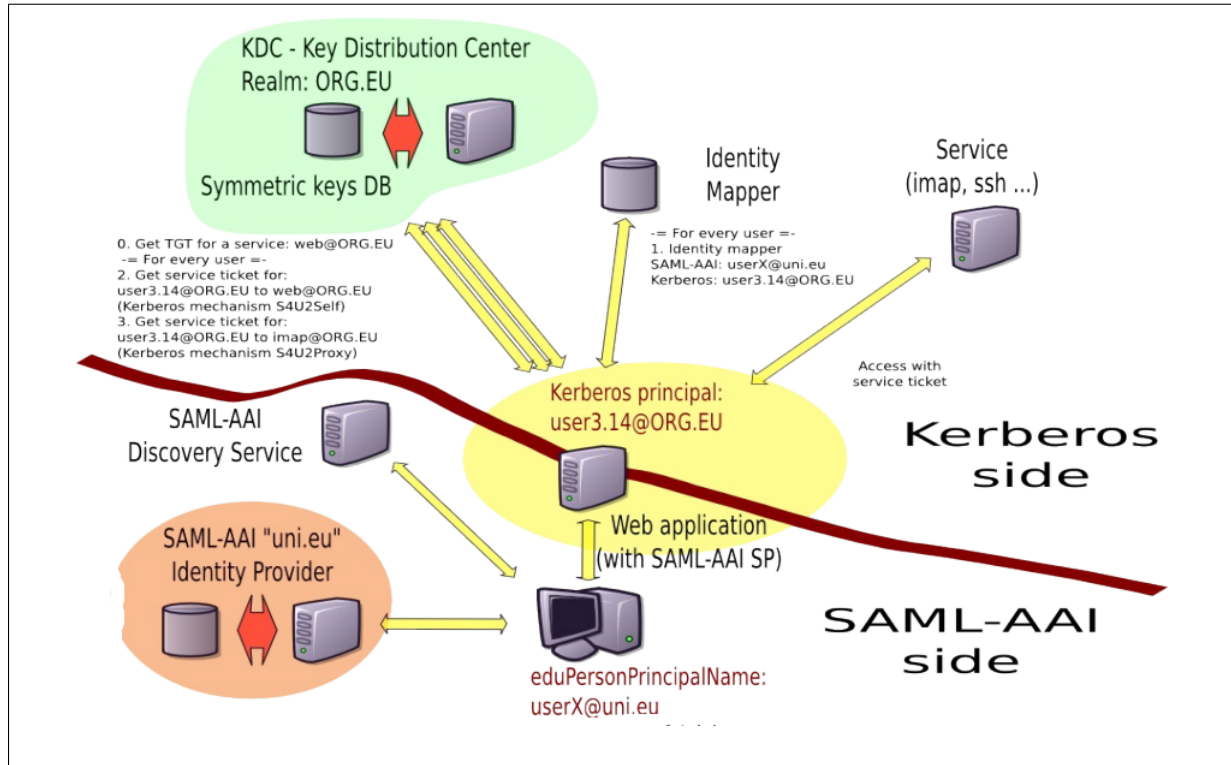


Figure 4.7: Interoperating SAML-AAI and Kerberos (Papez, 2009)

Kerberos single sign-on system requires that the end-user's identity to be registered in the same domain as the Kerberos key distribution centre (KDC) (whether registered in KDC or a database that can be accessed by KDC). It means that the end-user's identity from another domain can not be used for requesting ticket granting ticket. Papez (2009) proposes an identity mapping model in his research, it maps the end-user's identity to an identity that acceptable by the KDC (i.e. stored in the same domain as the KDC). For example, an end-user in domain "uni.eu" is accessing services in domain "org.eu" (figure 4.7). The end-user has the username "userX" in domain "org.eu", after federated authentication, the end-user's username is changed into "user3.14" in domain "org.eu".

In the approach of Papez (2009), the service provider requests Kerberos ticket granting ticket and service ticket on behalf of the end-user. This is achieved with Kerberos extension "Services For User to Self (S4U2Self)" and "Services for User to Proxy (S4U2Proxy)" (Microsoft, 2012a). S4U2Self and S4U2Proxy were introduced by Microsoft;

S4USelf allows a service to obtain a service ticket to itself on behalf of a user; S4U2proxy provides a service that obtains a service ticket to another service on behalf of a user. The resulting service ticket can be used for:

- Requesting service's own information,
- Accessing control local to the service's machine, impersonating the user,
- Requesting to some other service, impersonating the user.

With S4USelf and S4U2Proxy, the service providers request Kerberos service ticket on behalf of the end-users. The end-users request other service (in the same domain as the service provider) through the service provider. The service tickets are kept by the service provider (figure 4.7).

SAML-AAI/Kerberos presents the following advantage over the other approaches:

- It integrates with existing SAML based federated single sign-on for web-based applications.
- Applications that support Kerberos can use SAML-AAI/Kerberos.
- It supports both web-based and non-web based services/applications.

This approach, however, relies heavily on web-based user agents (i.e. web-browsers). Authentication for every application is done through web-browsers. No matter what service/applications (web-base or non-web based) is requested, end-users must access them with web-browsers. Service/applications creators need to create new web-based user clients for all of their existing non-web based applications. If the creators of these services/applications do not buy-in this model, the approach will fail.

Similar to Project Moonshot and SASL-SAML, desktop authentication is not considered by this model. For an end-user to access a web-browser, he/she usually needs to authenticate to a desktop first, therefore, two authentication processes are required for an end-user to access computer services/applications.

This approach can be incorporated into the KSS approach. SAML-AAI/Kerberos using web-based service to request Kerberos ticket in another domain, the same method can be employed in middle-ware systems. The middle-ware systems accept authentications from the end-users (using any user agents) and request Kerberos tickets for non-web based services/applications. Instead of using web-browsers, the end-users can use user agents that are correspondent to the computer services/applications. This way, application provider and network domains do not need to modify their existing services/applications.

## 4.5 Summary

The research reviews the existing FSSO technologies, it found that FSSO were initially developed for web-based services (e.g. Security Assertion Markup Language, Liberty Identity Federation Framework, Shibboleth and WS-Federated). While there are approaches aim to deploy FSSO to web-based and non-web based services (e.g. Project Moonshot, Kerberos Secure Sharing, SASL-SAML and SAML-AAI/Kerberos), they require modifications to network domains' authentication infrastructures and computer services/applications. If network domains and applications providers do not buy-in these models, they will fail.

The research suggests a new simplified federated single sign-on system that is simpler to deploy than existing approaches. It extends upon Kerberos Secure Sharing and SAML-AAI/Kerberos, and it assumes that network domains use Kerberos single sign-on for local network single sign-on and a federation (using SAML based federated single sign-on) has been established between these domains. Instead of extending Kerberos servers beyond network domains' boundary to support federated single sign-on, this system uses middle-ware approach to facilitate federated single sign-on. Next chapter presents the design of proposed approach and testing environment.

# Chapter 5

## The Design of a New Simplified FSSO System

### 5.1 Introduction

This chapter presents the design a new simplified federated single sign-on system and its testing environment. The design follows the suggestions from chapter 4, it extends upon Kerberos Secure Sharing and SAML-AAI/Kerberos. It aims to provide a system that can be deployed to network domains without modifying their existing authentication infrastructure and services/applications.

### Terminology and Conventions

Following are the terminologies and conventions used in the thesis:

- A principal is the basic entity that participates in authentication. In most cases a principal represents a user or an instantiation of a network service on a particular host. Each principal is uniquely named by its principal identifier.
- Encryption is the process of transforming data into a form that cannot be understood without applying a second transformation. The transformation is affected by an

encryption key in such a manner that the second transformation can only be applied by someone in possession of the corresponding decryption key.

- Plaintext is a message in its unencrypted form, either before the encryption transformation has been applied, or after the corresponding decryption transformation is complete. Ciphertext is the encrypted form as a message, the output of the encryption transformation.

Section 5.2 presents the assumptions that are required by the system design. Section 5.3 discusses the use of simulation instead of real world applications. Section 5.4 presents the goals of the new simplified federated single sign-on system. Section 5.5 presents the design environment. Section 5.6 specifies the requirements of the system. Section 5.7 presents three use cases of the new simplified federated singles sign-on system.

## 5.2 Research Assumptions

The new simplified federated single sign-on system is based on Kerberos Secure Sharing and SAML-AAI/Kerberos. It assumes the following:

1. A federation has been established between domains (i.e. trust has been built between them). SAML based federated single sign-on systems are used to authenticate end-users to access web-based services in different domains.
2. Each domain uses their own Kerberos single sign-on systems to authenticate their end-users.
3. Services/Applications and their correspondent user agents (web-based and non-web based) support authentications with Kerberos single sign-on systems.
4. Encryption technologies have been employed in the systems.

### 5.3 Simulation of Real World Applications

This chapter presents the design requirements of a new simplified federated single sign-on system. The design environment is developed as simulations of real world protocols and applications. This is due to the fact that the implementation of real world applications is extremely complex. A simulation based on the system design can present the life-cycle of a real world application in a clean manner. Since the simulations are developed based on real world designs, they will not lose their core functionalities.

### 5.4 Goals of a Simplified Federated Single Sign-On System

The aim of the federated single sign-on system is to allow end-users to access both web-based and non-web based services/applications with one federated single sign-on session. End-users authenticate once and are able to use diverse computer services (in multiple domains) without re-authentication.

In addition to the goal of federated single sign-on system, the goal of the new simplified federated single sign-on system is to be easier to deploy than SASL-SAML and SAML-AAI/Kerberos. It aims to improve upon the SAML-AAI/Kerberos model by eliminating the needs to create web-based user agents for every non-web based service application. It also minimises the modifications to the service applications and their respective user agents. This approach aims to have a simple deployment process, therefore, it is more acceptable by domains than SASL-SAML and SAML-AAI/Kerberos.

### 5.5 Environment Specification

To design the new simplified federated single sign-on system, the design environment needs to simulate the real world scenario. Federated single sign-on authenticates end-users across multiple domains, therefore the design environment must have two or more independent



domains. This research assumes that each domains support both SAML based federated single sign-on and Kerberos single sign-on, and this research aims to allow end-users to access both web-based and non-web based services. Therefore, the design environment must has the simulations of following components: Two independent network domains, web-based services, non-web based services, SAML based federated single sign-on system and Kerberos single sign-on system.

- **Two Independent Network Domains** - Every federated single sign-on process involves two network domains, the domain that provides identity and the domain that provides the computer services. The design environment specifies two independent network domains (Alpha and Beta). Domain Alpha (i.e. identity provider) provides end-users' identities. Domain Beta (i.e. service provider) provides computer services.
- **Web-based services** - The web-based service supports SAML based federated single sign-on. Its goal is to demonstrate web-based federated single sign-on, it is not required to provide any functionalities other than that.
- **Non-web based service** - The non-web based services/applications includes Unix/Linux desktops and Secure Shell. They use Kerberos server system for authentication. Their main goals are to demonstrate the local domain and cross domain authentication.
- **SAML based federated single sign-on system** - It support Kerberos single sign-on authentication system, it also provides identity assertions for end-users. It allows end-users in domain Alpha to access web-based services in domain Beta with one authentication session.
- **Kerberos single sign-on system** - It provides authentication for end-users to access services/applications that support Kerberos. An end-users authenticates once with Kerberos and can access other services/applications (that support Kerberos) without re-authentication.

## 5.6 Requirements Specification

The five components of the new simplified federated single sign-on system (Middle-ware system, Kerberos server system, SAML based federated single sign-on system, Web-based applications and non-web based applications) have the following requirements:

1. The middle-ware should provide functions for requesting, storing and sending the Kerberos tickets (i.e. ticket granting tickets and service tickets) on behalf of end-users. It should also support Kerberos single sign-on.
2. The non-web based applications simulate Unix/Linux Desktop and Secure Shell protocol. They should support Kerberos single sign-on system.
3. The simulation of Kerberos key distribution centre should provide functionalities to generate, verify and send Kerberos tickets (e.g. ticket granting tickets and services ticket).
4. The simulation of a web-based service application should support SAML based federated single sign-on.
5. The simulation of SAML based federated single sign-on system should provide web-based federated single sign-on.

## 5.7 Use Cases

This section presents several possible usage scenarios for the outputs of this research. Each use case uses the same network set-up. They contains two domains, Alpha and Beta. Federation has been established between domain Alpha and Beta using SAML based federated single sign-on system. Domain Alpha belongs to an academic institution (institution Alpha), it has numbers of end-users. It has an identity provider that manages the identities of end-user, “Chen”, “Fred” and “Paul”. Domain Alpha provides Unix/Linux desktop with limited processing power.

Domain Beta also belongs to an academic institution (institution Beta). It has two types of services, a remote Unix/Linux desktop with huge processing power that can be accessed through secure shell (SSH) connection and a web-based email service. It has a service provider that provides these computer services, and it does not manage any identity. The policy of the federation allows the end-users from domain Alpha to access computer services in domain Beta.

### 5.7.1 Use Case 1

End-user “Chen” login a Unix/Linux desktop in institution Alpha to do his lab work. The lab work requires Chen to compile a large amount of computer source codes. The lab time lasts one hour. The Unix/Linux desktop in institution Alpha does not have enough processing power to compile these source codes in this amount of time. If the source codes were not compiled in one hour, Chen will have to choose another time and abandon the current progress. Chen decides to use a remote Unix/Linux desktop in institution Beta for this task. He connects to the remote Unix/Linux desktop in domain Beta using the new simplified federated single sign-on system. The connection established immediately and did not require any authentication. Chen transfers all the source codes over to the remote desktop and the compilation finished under one hour. Chen retrieves the results and submits the lab work.

### 5.7.2 Use Case 2

Lecture “Fred” needs to set up 20 virtual machines (that running Linux) for students. These virtual machines are used for student to practise their Linux administration skills. The names of the students have been sent to Fred via email. Since Institution Alpha does not have a machine that is capable of hosting 20 virtual machines simultaneously, Fred needs to use computer resources in Institution Beta.

He first logs a Unix/Linux desktop in institution Alpha. Since institution Alpha does not provide any email services, Fred has been using the email service in institution

Beta. With the new simplified federated single sign-on system, Fred accesses the email and retrieves the names without re-authentication. He then connects to a remote Unix/Linux server in domain Beta (with the new simplified FSSO system) and setup virtual machines. The details of these virtual machines (e.g. IP address and domain name) are then sent to the students via email.

### 5.7.3 Use Case 3

End-user “Paul” needs a data storage machine to store mass amount of computer data, he also needs a high performance machines to process these data. A web interface (which is only accessible by authenticated end-users) is used to monitor the data processing. He uses the computer resources in Institution Beta to achieve this.

He first login into a local Unix/Linux desktop in institution Alpha. He then connects to the data storage machine and high performance machine in domain Beta using the new simplified federated single sign-on system. The process did not require any authentication on Paul’s part. He makes sure the data have been successfully transferred from the data storage to the high performance machine. He then accesses the web-based process monitor (with the new simplified FSSO system) and monitors the data processing. After data processing, he disconnects from all applications and systems.

## 5.8 Simplified Federated Single Sign-on System Lifecycle Description

This section covers three scenarios that may occur to an end-user. In subsection 5.8.1, an end-user in domain Alpha requests non-web based service application in domain Beta. In subsection 5.8.2, an end-user in domain Alpha requests web-based service application in domain Beta. In subsection 5.8.3, an end-user gained access to one type of service application (web-based or non-web based) and wishes to access another type.

### 5.8.1 Life-cycle 1: End-user accesses non-web base application

The middle-ware approach includes two ends, one resides in domain Alpha ( $M_\alpha$ ) and the other resides in domain Beta ( $M_\beta$ ). The two ends rely on the trust that was established between domain Alpha and Beta. In a scenario where an end-user from domain Alpha wants to access a non-web based service in domain Beta.  $M_\alpha$  and  $M_\beta$  handles the federated single sign-on processes (Figure 5.1). The end-user has an input and output relationship with the desktop system. They exchange information such as credentials, authentication results, application requests and applications request results.

Figure 5.1 highlights the components that are new. The components that have not been highlighted means they are un-modified components. The middle-wares ( $M_\alpha$  and  $M_\beta$ ) have been highlighted with dark background colour. It is the main contribution of this research. It facilitates federated single sign-on by requesting Kerberos ticket for end-users (in domain Alpha) in foreign domains (i.e. domain Beta). Network domains' Kerberos server and their computer services/applications do not need any modifications to work with the middle-ware system, they are not highlighted by any colour.

1. An end-user in domain Alpha log onto a desktop system by entering his/her credential. The credential will be sent to the Kerberos Key Distribution centre ( $K_\alpha$ ).
2.  $K_\alpha$  sends a search query to the end-user database in domain Alpha ( $D_\alpha$ ).
3.  $D_\alpha$  returns the results to  $K_\alpha$ .
4. If the credential was found in  $D_\alpha$ ,  $K_\alpha$  creates and sends a ticket granting ticket ( $TGT_\alpha$ ) to the desktop system. The  $TGT_\alpha$  will be stored in the desktop system.
5. The end-user requests a non-web based service application ( $A_\beta$ ) in domain Beta. The end-user accesses  $A_\beta$  with its respective user agent ( $U_\alpha$ ).
6.  $A_\beta$  sends a authentication request to  $U_\alpha$ .
7.  $U_\alpha$  presents the  $TGT_\alpha$  to  $K_\alpha$  and requests a service ticket ( $ST_\alpha$ ) for  $M_\alpha$ .

8. If  $TGT_\alpha$  is successfully verified by  $K_\alpha$ ,  $K_\alpha$  will create and send  $ST_\alpha$  to  $U_\alpha$ .  $ST_\alpha$  will be stored in  $U_\alpha$ .
9.  $U_\alpha$  presents  $ST_\alpha$  to  $M_\alpha$ .
10. If  $ST_\alpha$  is successfully verified by  $M_\alpha$ ,  $M_\alpha$  will create and send the identity assertion message to  $M_\beta$ .
11. Due to the trust between domain Alpha and Beta,  $M_\beta$  accepts the end-user's identity assertion. Then  $M_\beta$  generates a database query that will provision a new identity ( $I_\beta$ ) for the end-user in domain Beta. The query will be send to the end-user database ( $D_\beta$ ) in domain Beta.
12.  $D_\beta$  accepts the query and returns the result of identity provisioning to  $M_\beta$ .
13.  $M_\beta$  uses the  $I_\beta$  to authenticate to the KDC in domain Beta ( $K_\beta$ ) on behalf of the end-user.
14.  $K_\beta$  sends a search query to  $D_\beta$  to search for the  $I_\beta$ .
15. Since  $I_\beta$  has been provisioned in  $D_\beta$ ,  $D_\beta$  will find the credential and return the result to  $K_\beta$ .
16.  $K_\beta$  creates and sends  $TGT_\beta$  to  $M_\beta$ .  $TGT_\beta$  will be stored in  $M_\beta$ .
17.  $M_\beta$  sends  $TGT_\beta$  to  $K_\beta$  and requests  $ST_\beta$  for  $A_\beta$  on behalf of the end-user.
18.  $K_\beta$  verifies  $TGT_\beta$  and sends  $ST_\beta$  to  $M_\beta$ .
19.  $M_\beta$  sends  $ST_\beta$  to  $M_\alpha$ .
20.  $M_\alpha$  sends  $ST_\beta$  to  $U_\alpha$ .
21.  $U_\alpha$  presents  $ST_\beta$  to  $A_\beta$ .
22.  $A_\beta$  verifies  $ST_\beta$  and accepts the request from  $U_\alpha$ .

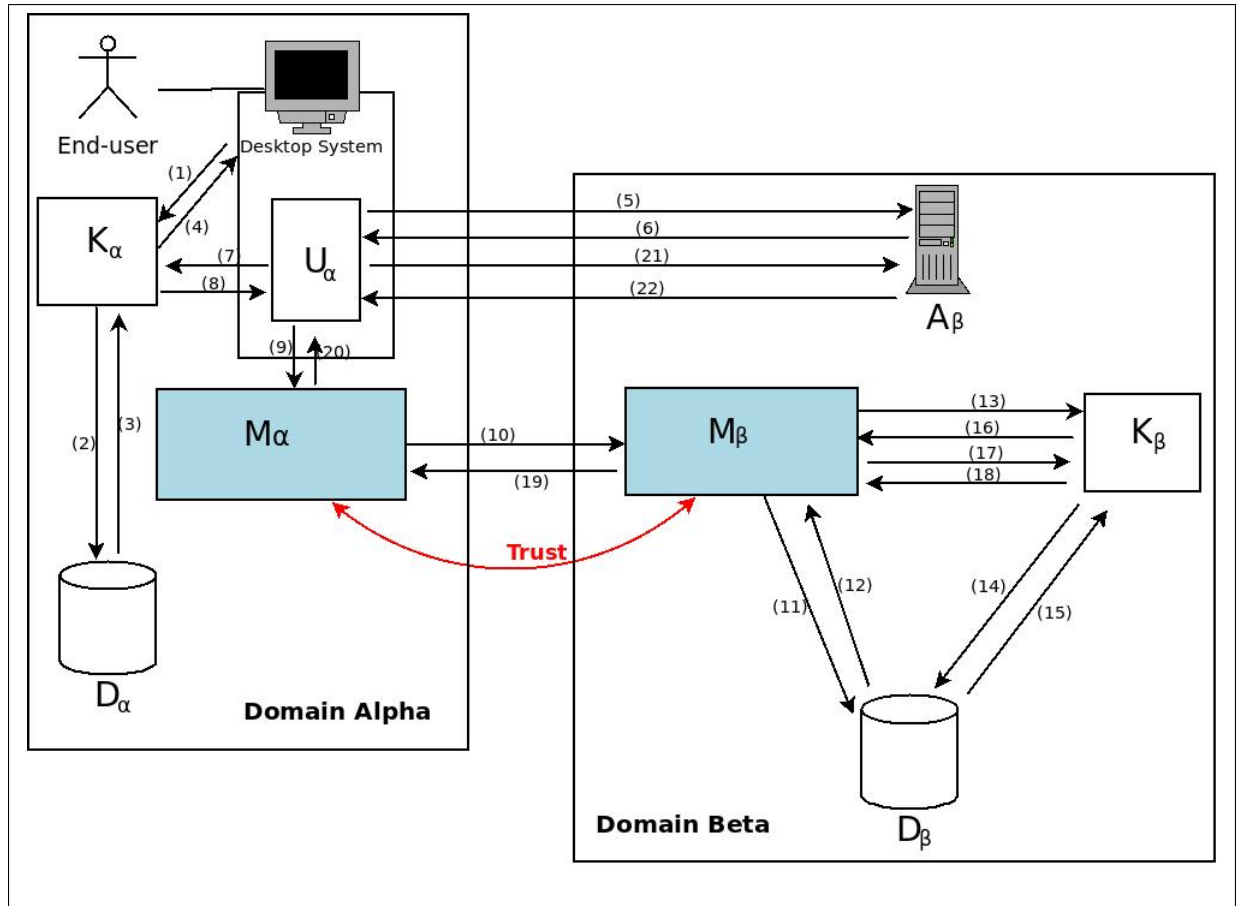


Figure 5.1: A New Simplified Federated Single Sign-on Life-cycle 1.

### 5.8.2 Life-cycle 2: End-user accesses web-based application

This subsection shows an end-user in domain Alpha requests web-based service application ( $SP_\beta$ ) in domain Beta. The user agent is a web browser ( $W_\alpha$ ) that runs on the desktop. The relationship between the end-user and desktop is the same as figure 5.1. They exchange information such as credentials, authentication results, application requests and applications request results.

1. The end-user log onto the desktop system by enter his/her credential. The credential is sent to the Kerberos key distribution centre ( $K_\alpha$ ).
2.  $K_\alpha$  creates and sends a search query to the database in domain Alpha ( $D_\alpha$ ).
3.  $D_\alpha$  returns the result of the search query to  $K_\alpha$ .
4. If the credential was found in the database,  $K_\alpha$  creates and sends the ticket granting ticket (TGT) to desktop system. The TGT will be saved on the desktop.

5. The end-user uses a web browser ( $W_\alpha$ ) to request a web-based service ( $SP_\beta$ ) in domain Beta.
6.  $SP_\beta$  requests  $W_\alpha$  to identify which domain the end-user belongs to.
7.  $W_\alpha$  re-directs to the identity provider ( $IdP_\alpha$ ) of domain Alpha.  $W_\alpha$  will retrieve the TGT from the desktop and send it to  $K_\alpha$ .
8.  $K_\alpha$  verifies the TGT and sends the service ticket ST to  $W_\alpha$ .
9.  $W_\alpha$  sends the ST to  $IdP_\alpha$ .
10. Once  $IdP_\alpha$  verifies the ST, it then creates an identity assertion message and passes it back to  $W_\alpha$ .
11.  $W_\alpha$  sends the identity assertion message to  $SP_\beta$ .
12.  $SP_\beta$  sends a search query to the end-user database in domain Beta ( $D_\beta$ ).
13.  $D_\beta$  returns the result to  $IdP_\alpha$ .
14. If a provisioned credential has been found in database,  $SP_\beta$  will grant the request from  $W_\alpha$ .

### 5.8.3 Other Life-cycles: End-user accesses web-based and non-web based applications

Section 5.8.1 and 5.8.2 present the life-cycle of two scenarios. These scenarios present the federated single sign-on life-cycles of one type of service applications (web-based or non-web based). We will continue to use the design environment that was described in design environment. This section discusses two more scenarios. First scenario is where an end-user from domain Alpha already gained access to non-web based service ( $A_\beta$ ) in domain Beta and wishes to access web-based services ( $SP_\beta$ ) in domain Beta. Second scenario is where an end-user from domain Alpha already gained access to  $SP_\beta$  and wishes to access  $A_\beta$ .



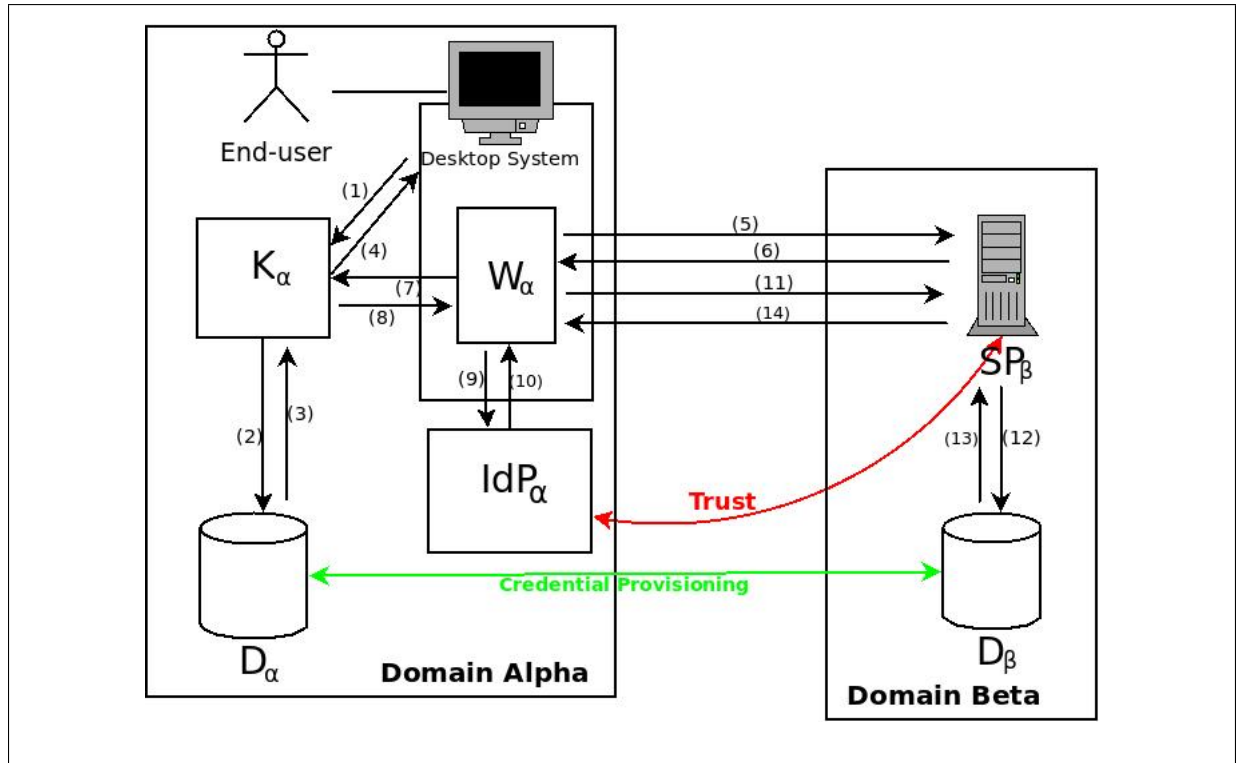


Figure 5.2: A New Simplified Federated Single Sign-on Life-cycle 2.

The two scenarios share the same desktop authentication process. The end-user needs to authenticate first to gain access to the desktop (figure 5.1 and 5.2).

1. The end-user enters his/her credential, it is then sent to the Kerberos key distribution centre ( $K_\alpha$ ) in domain Alpha.
2.  $K_\alpha$  generates a credential search query and sends it to the database ( $D_\alpha$ ) in domain Alpha.
3.  $D_\alpha$  executes the query and returns the result to  $K_\alpha$ .
4. If the credential was found in database,  $K_\alpha$  will return the result along with a ticket granting ticket to the desktop. The desktop will grant the access to the end-user.

In either scenario, the end-user needs to authenticate first to gain access to the desktop. The authentication results in the creation of a ticket granting ticket (TGT) which can be used to get service tickets (ST). The TGT is stored on the desktop. Therefore, in the first scenario, when an end-user gained access to  $A_\beta$ , a TGT is already created and stored on the desktop. So when the end-user accesses  $SP_\beta$ , he/she can use the same TGT for

authentication just like in life-cycle 2 (figure 5.2). The second scenario is similar to the first, the difference is the TGT is used to access to  $SP_\beta$  first, it was then used to access  $A_\beta$ .

## 5.9 New Simplified Federated Single Sign-on System Components

This section presents the component design of the new simplified federated single sign-on system. It consists of the design of network domains, end-user databases, SAML based federated single sign-on simulation (identity provider and service provider), federated single sign-on policy simulation, Kerbero single sign-on simulation, web-based service application simulation, non-web based service application simulation and the middle-ware system. All of the designs follow the requirement specifications.

### 5.9.1 Network Domain Design

As described in the requirement specification, the design environment contains two independent network domains. While describing the life-cycle of the new simplified federated single sign-on system, the two domains are assigned the name Alpha and Beta. Domain Alpha and Beta are used to distinguish two independent network domains in the life-cycle representation. In the component design, domain Alpha is assigned the domain name *home.virtual.vm*, and domain Beta is assigned the domain name *foreign.virtual.vm*. Their respective domain names *home.virtual.vm* and *foreign.virtual.vm* are used to implement the design environment (Table 5.1).

	<b>Domain Alpha</b>	<b>Domain Beta</b>
<b>Domain Name</b>	home.virtual.vm	foreign.virtual.vm
<b>Provide Service Applications</b>	No	Yes (web-based and non-web based)
<b>Provide end-user identities</b>	Yes	No

Table 5.1: Network Domain Names

### 5.9.2 End-user Database Design

The design of database aims to store information that support the authentication systems. It is designed to be used by the federated single sign-on systems and Kerberos single sign-on systems. They contain these basic information: username, password, role and domain name.

- Username, it is the unique identifier of the end-user. It represents an individual end-user. There should not be any repeating usernames in the database.
- Password, it is a combination of characters and numbers. It is the distinguishing characteristic that differentiates that particular end-user or group from others (Smith, 2001).
- Role, it represents a person's allotted share, part, or duty in life and society; the character, place, or status assigned to or assumed by a person (Oxford English Dictionary, 2010). In the context of this research, it represents an end-user's character, place and status in the network domain.
- Domain name, it is the name of the domain where the end-users reside in.

As presented in section 5.8, there are two independent databases in the system: database in domain Alpha ( $D_\alpha$ ) and database in domain Beta ( $D_\beta$ ). When a federation has been established between domains Alpha and domain Beta. The credentials in  $D_\alpha$  are provisioned in  $D_\beta$ . Subsection 5.9.2 presents variables in the database. Subsection 5.9.2 presents the provisioning of credentials. Subsection 5.9.2 presents the technology used to construct the databases.

#### Credential Provisioning

The research on identity provision is out of the scope of this research, therefore, his research manually provision the credentials in  $D_\alpha$  and  $D_\beta$ . Identity provisioning includes:

- Creating credentials in  $D_\alpha$ .

- Copying end-user's username from  $D_\alpha$  to  $D_\beta$ .
- Create new password for end-users in  $D_\beta$ . This is done to distinguish end-users' credentials in domain Alpha and their provisioned credentials in domain Beta.
- Mapping of the roles of end-users in domain Alpha into their correspondent roles in domain Beta. The mapping is based on the policies that were negotiated between two domains. The detailed policies are explained in subsection 5.9.3.
- Provision of domain name. End-users from domain Alpha will have the domain name *home.virtual.vm* in domain Alpha. The end-users' credentials that stores in  $D_\beta$  has the domain name *foreign.virtual.vm*.

## Database Variables

It was established that each credential contains these four database elements: Username, Password, Role and Domain name. This section describes the data types and variables of these elements.

### Variable Datatypes

Username, password, role and domain names are mixture of characters and numbers. *String* can represent a sequence of characters and integers. Therefore, this research chooses *string* as the datatype for these database elements.

### Variable Contents

This research creates three end-users and their credentials. Their usernames are *chen*, *fred*, and *paul* (Table 5.2). The end-users' passwords are different in domain Alpha and Beta. For example, end-user *chen* has the password "123" in domain Alpha, and the provisioned credential in domain Beta has the password "321".

Role represents an end-user's character, place and status in the network domain. The end-user *chen* has the role of "student" in domain Alpha. It means that *chen* has the role of "guest" in domain Beta. He can access service applications in domain Beta, however, he can not create, modify or delete the service applications. The end-users *fred*

and *paul* has the role of “lecture”. It means that they have the role of “power” in domain Beta, they can access, create, modify and delete the service applications in domain Beta.

Domain name is designed to be used by Kerberos Single Sign-On. Because of the symmetric nature of Kerberos protocol, and because Kerberos Key Distribution Centre (KDC) in domain Alpha and Beta are isolated, KDCs only authenticate end-users in their respective domains. Therefore, the provisioning of end-user credentials modifies the domain name from *home.virtual.vm* to *foreign.virtual.vm*.

	Domain Alpha	Domain Beta
<b>Username</b>	chen	chen
<b>Password</b>	123	321
<b>Role</b>	Student	Guest
<b>Domain Name</b>	home.virtual.vm	foreign.virtual.vm
<b>Username</b>	fred	fred
<b>Password</b>	123	321
<b>Role</b>	Lecture	Power
<b>Domain Name</b>	home.virtual.vm	foreign.virtual.vm
<b>Username</b>	paul	paul
<b>Password</b>	123	321
<b>Role</b>	Lecture	Power
<b>Domain Name</b>	home.virtual.vm	foreign.virtual.vm

Table 5.2: Domain Variables

## Database Technology

The Extensible Markup Language (XML) (Bray et al., 2008) has become the de facto standard for representing, storing, and exchanging electronic data (Zhang et al., 2011). It has been used to construct the end-users database. The end-users’ credential are stored in a structure manner with XML.

The database requires three entries (i.e. one for each end-user’s credential). This research creates XML tag <user> to mark each credentials (figure 5.3). Each credential contains four credential elements: username, password, role and domain name. Four XML tags are created <username>, <password>, <role> and <domain> to mark four credential elements. XML tag <users> is used to mark the name of the database.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<users>
  <user>
    <username>chen</username>
    <password>123</password>
    <role>student</role>
    <authenticated>true</authenticated>
    <domain>home.virtual.vm</domain>
  </user>
  <user>
    <username>fred</username>
    <password>123</password>
    <role>lecture</role>
    <authenticated>true</authenticated>
    <domain>home.virtual.vm</domain>
  </user>
  <user>
    <username>paul</username>
    <password>123</password>
    <role>lecture</role>
    <authenticated>true</authenticated>
    <domain>home.virtual.vm</domain>
  </user>
</users>
```

Figure 5.3: End-user Database Written in XML.

### 5.9.3 Federated Single Sign-On Policy Design

Federated single sign-on required the establishment of federation between domain Alpha and Beta. The federation is based on a set of policies. Although this is not the focus of this research, it is important to establish some basic policies. Chapter 4 provided an example of a security policy. Based on the example, this design establishes the following policies:

1. End-users from domain Alpha can access service applications in domain Beta.
2. Role “student” in domain Alpha has the role of “guest” in domain Beta. Guest can use services/applications and save contents on them, however, guest does not have the permission to administrate the service applications (i.e. create, modify and delete service applications).

3. Role “lecture” in domain Alpha has the role of “power” in domain Beta. End-users have the role “power” can access, create, modify and delete service applications in domain Beta.

Policy 1 is the basic policy of a federation. Identity provider in domain Alpha authenticates the end-user. Successful authentication means that the end-user is part of domain Alpha. Service provider in domain Beta accepts the end-user’s service request base on the 1st policy. Since the outcome is either true or false, it doesn’t need to be stored in a policy repository.

Policies 2 and 3 are more complex than policy 1. It maps the role of the end-user in domain Alpha to the role in domain Beta. A XML based policy repository is used in this research (figure 5.4). XML `<policy>` is used to indicate the root of the repository. `<student>` and `<lecture>` are the roles in domain Alpha. Under `<student>`, tag `<role>` has the value *guest*. Under `<lecture>`, tag `<role>` has the value *power*. This means an end-user, who has the role *student* in domain Alpha, has the role *guest* in domain Beta. An end-user, who has the role *lecture* in domain Alpha, has the role *power* in domain Beta.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<policy>
  <student>
    <role>guest</role>
  </student>
  <lecture>
    <role>power</role>
  </lecture>
</policy>
```

Figure 5.4: Federation Policies Written in XML.

### 5.9.4 Kerberos Single Sign-On Simulation Design

This research assumes the establishment of Kerberos single sign-on system in each domain. This means that domain Alpha and Beta implemented Kerberos single sign-on system as

their authentication mechanism. In addition, the service applications (web-based and non-web based) and their respective user agents support Kerberos single sign-on systems. To simulate this environment, this research designs a Kerberos single sign-on system simulation.

Encryption of data (e.g. TGTs, STs, credentials and shared keys) is a primary focus of Kerberos single sign-on system. Storing and exchanging data employee data encryption. This research, however, does not focus on the encryption aspect of Kerberos. It assumes that encryptions are applied during data storage and exchange.

### **Kerberos Single Sign-On Simulation Goals**

The simulation follows the literature review (section 3.3.3) on Kerberos single sign-on system. A Kerberos system contains a key distribution centre (KDC) and ticket granting server (TGS). According to Kohl et al. (1991), KDC and TGS reside in the same physical server. They have access to the same end-user database. The differences between KDC and TGS are their functions. KDC authenticates the end-users' credentials and issues ticket granting tickets (TGT). KGS verifies the TGT and issues service tickets (ST).

The research designs a Kerberos system simulation that includes the functions of KDC and TGS. Since the differences between them are logical, they are merged into a single component with all the core functions of KDC and TGS. The merge of KDC and TGS keeps their core functions, therefore, it should not affect the outcome of the research. The Kerberos system simulation includes the following core functions:

- Authenticate end-user's credential.
- Create ticket granting ticket (simulate KDC).
- Create shared key between service application and Kerberos server.
- Verify ticket granting ticket.
- Create service ticket.



- Create and maintain single sign-on session.

In addition to the core functions, Kerberos system simulation requires the following:

- Service daemon that accepts client connections.
- Connect and extract data from end-user database (section 5.9.2).

### **Kerberos Server Daemon**

The Kerberos system authenticates the end-users and issues ticket granting tickets and service tickets. It also issues shared keys to services. This requires a static IP address and a static port for accepting incoming requests. According to IANA (2012), dynamic and/or private ports are from 49152 to 65535. This research chooses port 50000 as the port of Kerberos system.

The Kerberos system simulation uses network sockets for network communication. Sockets allow applications to communicate using standard mechanisms built into network hardware and operating systems. It is created with the identified IP address and network port. This research uses transmission control protocol (TCP) as the socket protocol. TCP employs error detection and provides reliable network communications.

### **Database Functions**

Extracting the credential data from the database is one of the tasks of the Kerberos system simulation. This allows Kerberos system simulation to authenticate the end-user by comparing the credential information in the database and the credential information of the end-user. Section 5.3 describes an end-user database that was constructed with XML language (figure 5.3). The end-user database stores the credential data (e.g. username, password, role and domain name) of the end-users. These data were manually organised and entered into the database (section 5.9.2). The database is stored in a form of a XML file. Therefore, the database functions include the following:

- Reading the entire content of the XML file.
- Searching through the content to find the credential data.

Since the content of the database is organised in a tree format (figure 5.5). The searching function starts from the root element of the tree (i.e. `<users>`) and travel to child element (i.e. `<user>`) and sub child elements (i.e. `<username>`, `<password>`, `<role>` and `<domain name>`).

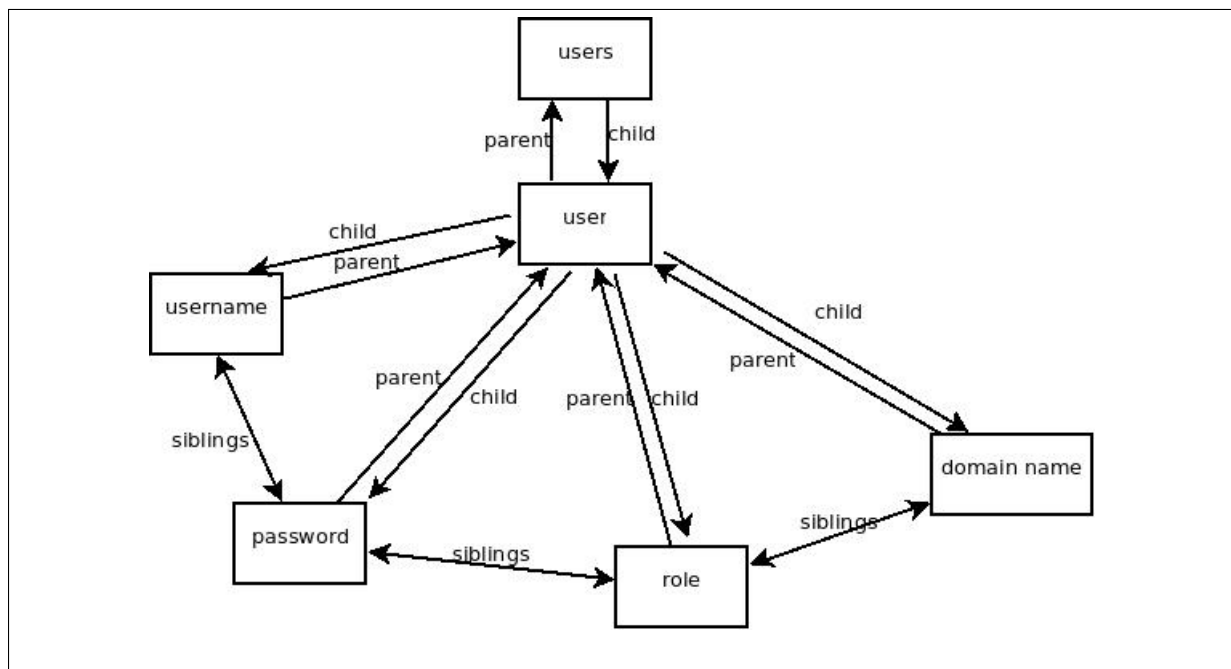


Figure 5.5: Database Tree Relation.

`<users>` is at the top of the hierarchy (figure 5.6); it contains all the identity information; it's the first to be accessed in the database. `<user>` is the child element; it contains the identity information of a single end-user. The contents between `<user>` and `</user>` represent a single end-user.

### Authentication Function

Authentication is one of the core functions of Kerberos system. It ensures that the end-users are who they claim to be. The design of authentication function is based on the database functions. It involves two elements in the database: username and password.

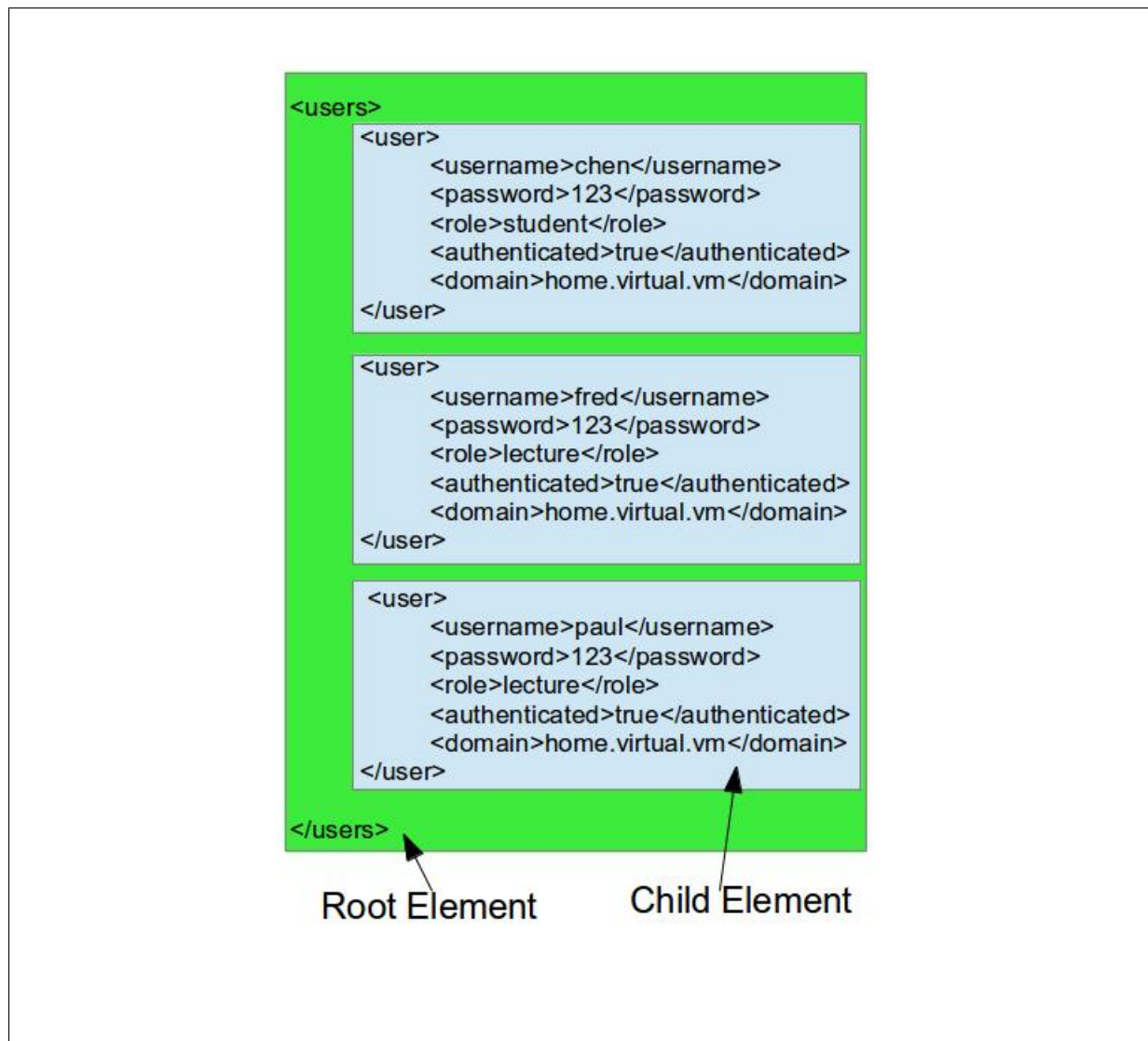


Figure 5.6: Database Tree Relation in Colour.

The database function travels from root element to child element. In this process, the authentication function aims to match the credential data (i.e. username and password) presented by the end-user and the credential data in the database. If a match was found, the authentication function returns a success message to the end-user and the Kerberos server simulation.

### Create and Verify Ticket Granting Ticket

Ticket granting ticket (TGT) is one of the core component of Kerberos system. It shows that the holder of the TGT has been successfully authenticated by the Kerberos system. This research designs a simulation of the TGT based on Needham & Schroeder (1978);

Kohl et al. (1991). According to Needham & Schroeder (1978); Kohl et al. (1991), TGT contains the shared key between the Kerberos server and the end-user and expired time.

The simulation of ticket granting ticket is a string that merges the username, password, role, authentication result, domain name, issues time, expired time and hash key. They are separated by a semicolon (figure 5.7). The hash key is a SHA-2 hash of the source strings. SHA-2 hash function generates a single fixed length string from the source. The source can be any types of data (in this case, the source strings). The generated hash is appended to the end of the ticket granting ticket.

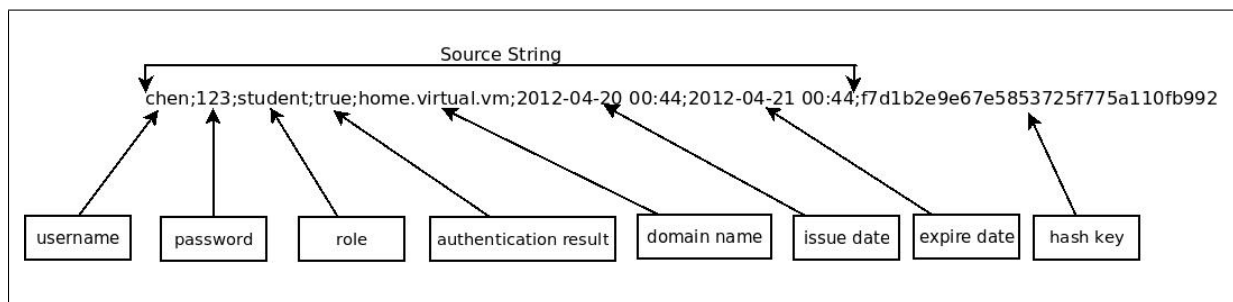


Figure 5.7: Simulate Kerberos Ticket Granting Ticket.

Kerberos system simulation saves a copy of the TGT on the server. The copy is saved in the form of text file; it's named after the end-user's username. The original TGT is sent to the end-user in the form of plaintext. The end-user saves the TGT in a text file for later use (e.g. obtaining service ticket).

### Create Shared Key between Services and Kerberos Server

A shared key between Kerberos server simulation and service simulations has two goals: It shows that the service has been registered in the Kerberos server simulation, and it can be used to verify the end-users' service tickets.

A service simulation is manually registered in the service database. The database is constructed using XML language. The variables are stored in a tree format that similar to the end-user database (figure 5.8). The root element is `<services>`. `<service>` represents a single service. `<serviceName>` represents the name of the service simulation. `<domain>` represents the domain name of the service simulation. `<ipAddress>`

represents the IP address of the service simulation. `<services>` is the parent element of `<service>`. One `<services>` contains multiple child elements (i.e. `<service>`). `<service>` is the parent element of `<serviceName>`, `<domain>` and `<ipAddress>`. In the sample (figure 5.8), the service name is “ssh”, it represent a secure shell server (described in section 5.9.5). The sample domain name of the ssh server is “home.virtual.vm”. The sample IP address is “127.0.0.1”.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<services>
  <service>
    <serviceName>ssh</serviceName>
    <domain>home.virtual.vm</domain>
    <ipAddress>127.0.0.1</ipAddress>
  </service>
</services>
```

Figure 5.8: Sample Service Database Written in XML.

To obtain a shared key, Kerberos server simulation needs to verify the service simulation. It receives the request for shared key, along with the service simulation’s service name, domain name and IP address. Kerberos server simulation verifies the service simulation by searching for the matching content in the service database. Kerberos server simulation travel from the root element `<services>` to child element `<service>`; it compares the content of `<serviceName>` and `<domain>` with the ones provided by the service simulation. If the service simulation was registered in the service database, a matching pair should be found, the Kerberos server simulation will then create a shared key for the service provider. If the service simulation was not registered in the service database, a matching pair should not be found. The Kerberos server simulation should not create any shared key for the service simulation.

Creating a shared ticket requires three elements: service name, domain name and IP address. They are presented as strings. They are merged into a single string (source string) and only separated by semicolons (figure 5.9). This source string is used to create shared key.

This research uses SHA-2 hash function to create shared keys between Kerberos

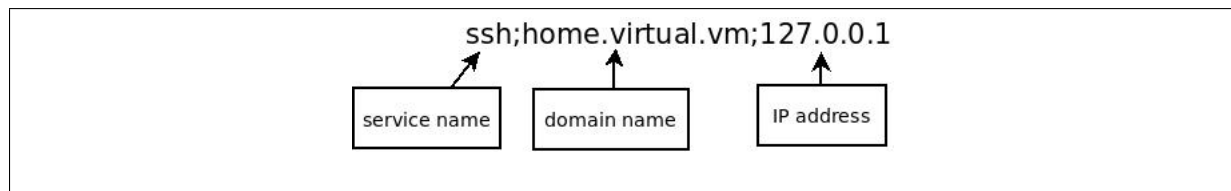


Figure 5.9: Sample Source String.

server and the services. A string is created by putting the source string through SHA-2 hash function. The string is appended to the back of the source string (figure 5.10), the result is then sent to the service simulation in the form of plaintext. The service simulation stores the shared key in the form of a text file.

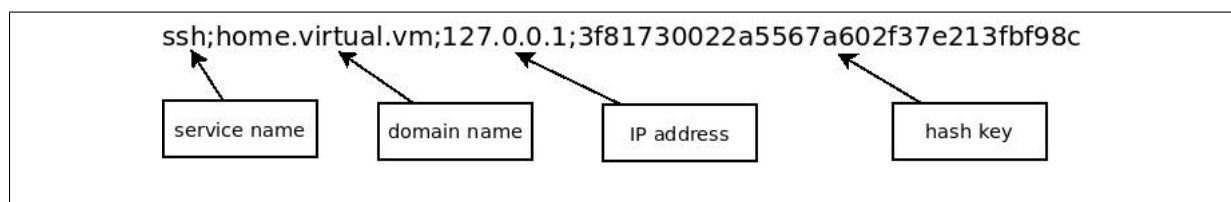


Figure 5.10: Sample Shared Key.

### Create Service Ticket

To obtain a service ticket (ST), the end-user's ticket granting ticket (TGT) needs to be verified by the Kerberos system simulation. The end-user's user agent reads the content of ticket granting ticket (from the text file) and sends the TGT content (in the form of plaintext) to the Kerberos server simulation. It also sends the name of the requested service to the Kerberos server simulation. Upon receiving the ticket granting ticket, Kerberos server simulation searches for the correspondent copy on the server. If the copy was not found, the end-user's request will be rejected. If the copy was found, Kerberos server will continue with the TGT verification process.

The ticket granting ticket (TGT) verification processes include the verification of username, domain name and hash key. In addition, it verifies that the issues date has not reach the expired date. Kerberos server simulation extracts these variables from the TGT and the correspondent copy. If the username, domain name and hash key in the TGT matched the correspondent variables in the copy, and if issue date on the TGT does not exceed the expired date on the copy, the verification is a success. Else, the end-user's

request will be rejected.

If Kerberos server simulation successfully verifies the ticket granting ticket, it will then create and send a service ticket to the end-user. The creation of service ticket is the same as creating the shared key between a server and Kerberos server simulation. The Kerberos server simulations access the service database and uses the service name that requested by the end-user to search for the service data (e.g. service name, domain name and IP address). These service data are merged into a single string (source string), semicolons are used to separate the individual data. A SHA-2 hash is created by putting the source string through SHA-2 hash function (figure 5.11). The hash key is the service ticket; it is sent to the end-user's user agent.

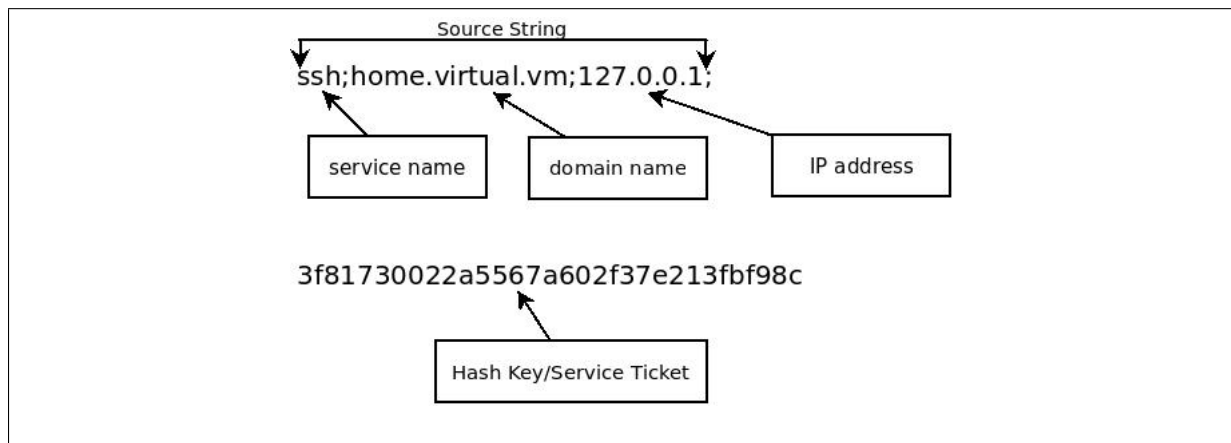


Figure 5.11: Sample Service Ticket.

### 5.9.5 Non-web Based Service Application Simulation Design

The goals of the non-web based service simulations are used to demonstrate the Kerberos server simulation and the new simplified federated single sign-on system. As described in section 5.2, this research assumes the non-web based services/applications and their respective user agents support Kerberos single sign-on. Therefore, the simulations need to simulate applications that support Kerberos single sign-on system.

According to section 3.3.3, Unix/Linux shell and secure shell protocol support Kerberos single sign-on system. Unix/Linux shell provides interface for end-users to interact with the desktop system. A simulation of Unix/Linux desktop and shell is designed in

this research. A simulation of secure shell protocol is also designed in this research, it provides connection between two Unix/Linux shell simulations. It allows end-users to access multiple Unix/Linux shell simulations at different network locations.

This requirement specified two domains. For demonstration, domain Alpha and Beta contain the same set up of Unix/Linux shell and SSH simulation (figure 5.12). SSH client and server simulations run on top of the Unix/Linux shell simulation. The end-user accesses the SSH client through Unix/Linux Shell. Relationship 1 represents the exchanges of message between the end-user and the Unix/Linux shell simulation; relationship 2 represents the exchanges of information between the SSH client simulation and the SSH server simulation.

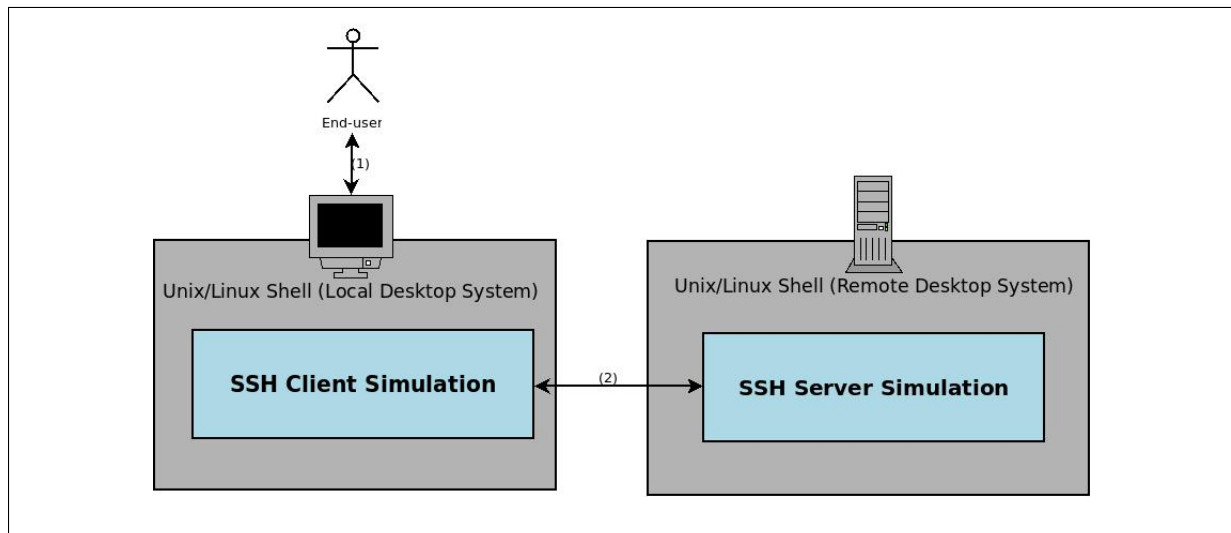


Figure 5.12: Unix/Linux Shell and SSH simulation.

### Unix/Linux Desktop Simulation Design

According to Newham & Rosenblatt (2005), “Shell” is the Unix/Linux term for a user interface to the system – something that lets you communicate with the computer via the keyboard and the display. The Unix/Linux shell allows end-users to interact with the Unix/Linux desktop system. It allows end-users to create, modify, delete or execute files. These files can be any type (e.g. text files). Unix/Linux shell also supports file directories (or folders) – virtual containers that used to organises files. It allows end-users to create, modify and delete file directories.



User agent such as SSH client runs on top of Unix/Linux desktop. It is evoked through Unix/Linux shell. In addition, Unix/Linux desktop requires the end-users to login before accessing the Unix/Linux shell. This research designs the simulation of Unix/Linux desktop and shell. The design of Unix/Linux desktop simulation login includes the following components:

- **Connecting to Kerberos Server Simulation**

Unix/Linux desktop simulation creates a network socket and uses it to establish connection to the Kerberos server simulation. The connection allows the Unix/Linux desktop simulation to send ticket granting ticket requests to the Kerberos server simulation.

- **Present login prompt to the end-user**

The Unix/Linux desktop simulation presents a login prompt that asks for a username and a password. The login prompt asks for the end-user's username first, then it asks for the end-user's password. It will display the username as it been entered; it will not display the password.

- **Requesting and Storing Ticket Granting Ticket**

The simulation sends the pair of username and password to the Kerberos server simulation along with a request for a ticket granting ticket. Section 5.9.4 describes the credential verification process. If the Kerberos server simulation successfully verifies the credential, a ticket granting ticket will be created and returned to the Unix/Linux shell simulation.

- **Storing Ticket Granting Ticket**

Upon receiving the ticket granting ticket, the Unix/Linux desktop simulation accepts the end-user's access and presents it with a Unix/Linux shell simulation. The ticket granting ticket (TGT) is stored on the desktop simulation in the form of a text file. The TGT can be access by user agents such as SSH client simulation.

After an end-user is successfully authenticated, the Unix/Linux shell simulation

provides a list of commands. These commands simulate the real world Unix/Linux commands. Their goal is to distinguish the different Unix/Linux desktop simulations. Since there are multiple Unix/Linux desktop simulations running at the same time (figure 5.12), it is important to distinguish them. The execution of these commands returns different results on different Unix/Linux desktop simulations. The results of these commands determine which desktop simulation the end-user is currently accessing. Following is the list of functions (Unix/Linux command name at the beginning):

- “ls” - Listing the contents (files and directories) in the current directory.
- “touch” - Create a text file.
- “hostname” - Display the hostname of the current Unix/Linux desktop simulation.
- “ifconfig” - Display the IP address of the current Unix/Linux desktop simulation.
- “klist” - Listing Kerberos ticket granting tickets on the current Unix/Linux desktop simulation.
- “ssh” - Secure shell protocol simulation.
- “exit” - Exiting the system. Termination of simulations will not remove any saved files.

### **Secure Shell Protocol Simulation Design**

According to Ylonen & Lonvick (2006), the Secure Shell Protocol (SSH) is a protocol for secure remote login and other secure network services over an insecure network. SSH allows end-users to authenticate with Kerberos authentication system (MIT, 2008; Migeon, 2008). The research designs a SSH simulation that allows end-users to access remote Unix/Linux desktop simulations from their local Unix/Linux desktop simulations. The SSH simulation contains two ends, client and server. The SSH client is a user agent runs on the end-user’s desktop (i.e. Local Unix/Linux desktop simulation). It initials the SSH connection by sending requests to the SSH server. The SSH server is a software daemon

that runs on the remote Unix/Linux desktop simulation, it listens for SSH requests from SSH clients. The SSH client and server maintain the SSH session. The end-users evoke the command of the remote Unix/Linux server simulation by sending requests through the SSH session.

The goal of a SSH simulation is to demonstrate the authentication system of Kerberos server simulation and the new simplified federated single sign-on system. This research assumes the connection between the SSH client and server is secure (i.e. Network encryption is applied). The SSH simulation supports the Kerberos server simulation, so that the end-users can authenticate to the remote SSH server simulation in a single sign-on fashion.

### **Secure Shell Client Simulation Design**

SSH client uses network sockets for transporting message. The network sockets is created with the IP address and network port number. The IP address and port belongs to the remote server (e.g. Kerberos server simulation and SSH server simulation). The network socket can request, send and receive network package. All the messages were sent as plain text. This research assumes that the network is encrypted.

When end-user accesses a remote SSH server simulation that is in the same domain as the end-user. The SSH client simulation has the following functions:

- Extracting ticket granting ticket information (TGT) from the TGT text file.

The ticket granting ticket created by the Kerberos server simulation is stored on the Unix/Linux desktop simulation. It is stored in the form of a text file. Secure shell client simulation has the function that reads the text file and extracts the content into a string. The string is stored in the SSH client simulation's memory.

- Sending the ticket granting ticket information (with the request for service ticket) to the Kerberos server simulation.

The SSH client simulation uses network socket to connect to the Kerberos server simulation. Network socket allows the client to send the ticket granting ticket string to the Kerberos server simulation.

- Receiving the service ticket from the Kerberos server simulation.

Kerberos server simulation returns the service ticket to the client simulation. The service ticket creation process is described in section 5.9.4. The service ticket is sent through the network connection in the form of plain-text.

- Sending the service ticket to the SSH server simulation

The SSH server simulation runs on a remote Unix/Linux desktop simulation. Another network socket is required to establish connection between the SSH client simulation and the server simulation. After establishing the network connection, SSH client simulation sends the service ticket to the SSH server simulation. The service ticket is sent in the form of plain-text.

- Receiving the result of service ticket verification.

The SSH client simulation receives the result of service ticket verification. If the verification was successful, maintaining the network socket to maintain the SSH session. If the verification was unsuccessful, closing the network socket to terminate the SSH session.

- Send the end-user's command requests to the SSH server simulation.

The Unix/Linux shell simulation allows the end-user to interact with the Unix/Linux desktop simulation. In the SSH session, instead of executing the end-user's commands on the local Unix/Linux desktop simulation, the commands are executed on the remote Unix/Linux desktop simulation. The Unix/Linux shell simulation receives the commands from the end-user; the SSH client simulation sends these commands to the SSH server simulation. The remote Unix/Linux desktop simulation, which runs the SSH server simulation, executes the commands. The results of the executions are sent back to the SSH client simulation from the SSH server simulation.

- Receiving the result from the SSH server simulation.

The SSH client receives the result of executing the commands; the Unix/Linux shell simulations displays them to the end-user.

The SSH client is in domain Alpha, while the SSH server simulation is in domain Beta. When an end-user requests access to a SSH server that is in domain Beta, a service ticket of domain Beta is required. This SSH server simulation can not verify the service ticket that was created by a Kerberos server simulation of domain Alpha. This research designs a middle-ware system that allows end-users to access non-web based service/applications in a federated single sign-on fashion. The middle-ware system contains two ends, an identity provider and a service provider. The identity provider is in domain Alpha; the service provider is in domain Beta. The detail processes of them are described in section 5.9.8. The SSH client simulation has the following functions:

- Requesting service ticket for accessing the middle-ware system's identity provider.  
The SSH client simulation can not request a service ticket for SSH server simulation in domain Beta, instead, the SSH client simulation requests a service ticket for the middle-ware system's identity provider.
- Receiving the service ticket from Kerberos server simulation.  
Once the ticket granting ticket is verified by the Kerberos server simulation, a service ticket is created, it allows the end-user to access the middle-ware system's identity provider. The service ticket is sent to the SSH client simulation. SSH client receives and keeps the service ticket.
- Sending the service ticket to the middle-ware system's identity provider.  
SSH client simulation sends the service ticket to the middle-ware system's identity provider.
- Receiving the service ticket from the middle-ware systems identity provider.  
The service ticket that allows the end-user to access SSH server simulation in domain Beta is created; it is sent to the SSH client simulation by middle-ware system's identity provider. The SSH client simulation receives the service ticket and sends it to the SSH server simulation.

Once the SSH server simulation in domain Beta verifies the service ticket, it accepts the

service request from the end-user. The end-user can send commands through SSH client simulation to the SSH server simulation.

### Secure Shell Server Simulation Design

The SSH server simulation creates a network socket that listens to connection requests. It listens on a specific network port, and it can handle multiple connection at the same time. The SSH server simulation has the following functions:

- Requesting shared key from the Kerberos server simulation.

A shared key (described in section 5.9.4) between SSH server and Kerberos server simulation is needed to verify end-user's service tickets. To obtain the shared key, SSH server needs to be registered in simulation's service database. The SSH server has been manually registered in the service database under the name "ssh" (figure 5.13). The domain name is "home.virtual.vm". The IP address is a sample IP address, it will be re-assigned later.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<services>
  <service>
    <serviceName>ssh</serviceName>
    <domain>home.virtual.vm</domain>
    <ipAddress>127.0.0.1</ipAddress>
  </service>
</services>
```

Figure 5.13: Sample SSH server simulation Entry in the Service Database.

The SSH server uses network socket to connect to the Kerberos server simulation. Following the processes described in section 5.9.4, the SSH server sends its service name, domain name and IP address to the Kerberos server simulation. Since SSH service has been registered in the service database, it should obtain a shared key from the Kerberos server simulation. The shared key consists of the service name, domain name, IP address (i.e. sample IP address) and hash key. They are merged into a single string and separated by semicolon (figure 5.14).

- Verifying the end-user's service ticket.

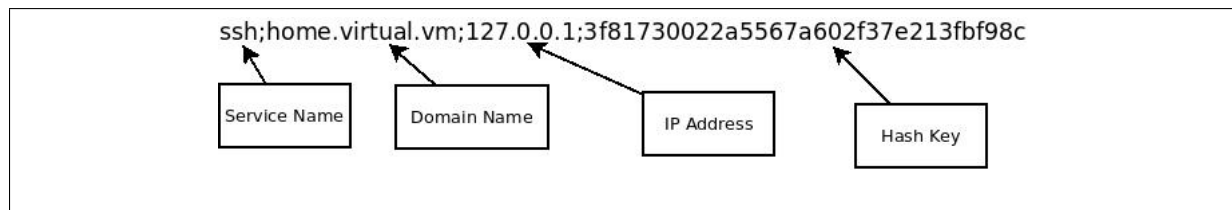


Figure 5.14: Sample Shared Key between SSH Server and Kerberos Server Simulation.

The SSH client simulation sends a message that contains the service ticket and the end-user's username to the SSH server simulation. The service ticket is appended to the username, and the two strings are separated by a semicolon (figure 5.15). The SSH server simulation compares the hash key in the shared key and the service ticket.

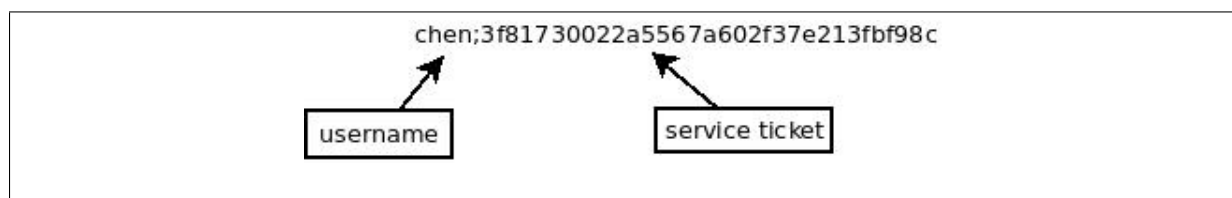


Figure 5.15: Sample Message that includes a Username and Service Ticket.

- Maintaining the network socket to maintain the SSH session.

If these two match, the end-user's service ticket is successfully verified. The SSH server maintains the network socket that is connecting to the SSH client simulation. It receives the command requests from the SSH client simulation; the Unix/Linux desktop simulation, which runs the SSH serve simulation, executes these commands. The SSH server simulations returns the results to the SSH client simulation.

- Closing the connection.

If service ticket was NOT verified, the SSH server simulation returns the result and closes the network socket. Closing the network socket results in the termination of SSH session.

### Exiting the SSH session

SSH allows end-users to access remote Unix/Linux desktop simulations. SSH simulation runs on top of the Unix/Linux desktop simulation, however, they are isolated sessions. The termination of SSH session will not terminate the (local or remote) Unix/Linux

desktop session. Once the end-user issues the “exit” command in a SSH session, the SSH client simulation will send a termination message to the remote SSH server simulation. The SSH server confirms the termination, both side will terminate the connection by closing their network sockets. The termination of SSH session ends the end-user’s access to remote Unix/Linux desktop and returns the end-user to its local Unix/Linux desktop simulation.

Similar to local desktop, termination of SSH session will not remove the end-user’s saved files on the remote Unix/Linux desktop simulation. The text files and directory can be access again in the next connection.

### 5.9.6 SAML based FSSO Simulation Design (Identity Provider)

Federated single sign-on model includes three components, user agent, identity provider and service provider. User agent is the client program (in this case, a web-browser) that is used by the end-user to access web-based services. Identity provider (IdP) manages authentication related tasks (e.g. web-based authentication and identity assertion). Service provider (SP) provides web-based services.

This section presents the design of the simulation the identity provider component. Its goal is to simulate the process of authenticating end-users (using web portal) and creating identity assertions. In addition, the identity provider supports Kerberos single sign-on, therefore, the simulation supports Kerberos server simulation (described in section 5.9.4). Section 5.9.7 focus on the design of service provider.

As described in section 5.8, a federation was built between domain Alpha and Beta, and the policies allow the end-users from domain Alpha to access computer service in domain Beta. Domain Alpha manages the identities of end-users, therefore, the identity provider resides in domain Alpha. Domain Beta provides computer services, therefore, the service provider resides in domain Beta.



## **Username/password Authentication**

In chapter 3, section 3.2.1 described the username/password authentication model. The end-user presents the username (an unique identifier of the end-user) and password (a secure code that was agreed between the end-user and the system) to the identity provider simulation. The authentication processes consist of identity provider simulation accessing the end-user database and searching for the matching pair of username and password.

As described in section 5.9.2, there are two end-user databases in the design. The database in domain Alpha contains the identity informations of the end-users; the database in domain Beta contains the identities that were provisioned from domain Alpha. The identity provider in domain Alpha only has access to the end-user database in domain Alpha.

The identity provider simulation presents a web-based authentication portal to the end-user. The portal consists of two text boxes, one for accepting username and one for accepting password (figure 5.16). It also includes two buttons, one for submitting the data (username and password) and one for resetting the text boxes. The username is displayed and entered into the system in the form of plain-text. The password is entered in the form of plain-text, however, it's not displayed.



The figure shows a web-based authentication portal design. It consists of a rectangular container with a light gray border. Inside the container, the text "Username:" is followed by a text input box. Below this, the text "Password:" is followed by another text input box. At the bottom of the container, there are two buttons: "Login" and "Reset".

Figure 5.16: Username/Password Portal Design.

## **Authenticating the Credential**

Once the end-user entered its username and password, the identity provider searches a matching pair in the end-user database. Similar to Kerberos authentication, the identity provider simulation contains functions that searches the end-user databases for matching username and password. The functions read the database and travel from the root element to the child elements.

For every sub-child element, the authentication function looks for the element tag username and the element tag password. It uses string comparing method to compare the end-user's credential with the ones stored in the database. If a pair of username and password in the database matches the one presented by the end-user, the authentication is a success. If the end-user's username and password do not match any pair in the database, the authentication is a failure. The identity provider simulation only provides identity assertion for authenticated end-users.

### **Support of Kerberos Single Sign-On**

The research assumes that the identity provider support Kerberos single sign-on. Therefore, the identity provider simulation is designed to support Kerberos server simulation. Section 5.9.4 presents a design of Kerberos server simulation. It simulates the functions of Kerberos key distribution centre and ticket granting server. In the Kerberos server simulation, an end-user needs to be authenticated first before accessing the Unix/Linux desktop simulation. Since web-browser can only be accessed after the end-user login the desktop, we can assume that an end-user already authenticated by Kerberos server simulation when accessing a web-browser.

#### **Requesting Shared Key**

A shared key (described in section 5.9.4) between identity provider simulation and Kerberos server simulation is required to verify end-user's service tickets. To obtain the shared key, identity provider simulation needs to be registered in the service database. The identity provider simulation is manually registered in the service database under the name "idp" (figure 5.17). The domain name is "home.virtual.vm". The IP address is a sample IP address; it will be re-assigned later.

The identity provider simulation uses network socket to connect to the Kerberos server simulation. Following the processes described in section 5.9.4, it sends its service name, domain name and IP address to the Kerberos server simulation. Since identity provider simulation has been registered in the service database. It should obtain the

```
<?xml version="1.0" encoding="iso-8859-1"?>
<services>
  <service>
    <serviceName>idp</serviceName>
    <domain>home.virtual.vm</domain>
    <ipAddress>127.0.0.1</ipAddress>
  </service>
</services>
```

Figure 5.17: Sample Identity Provider Entry in the Service Database.

shared key from the Kerberos server simulation. The shared key consists of the service name, domain name, IP address (i.e. sample IP address) and hash key. They are merged into a single string and separated by semicolon (figure 5.18).

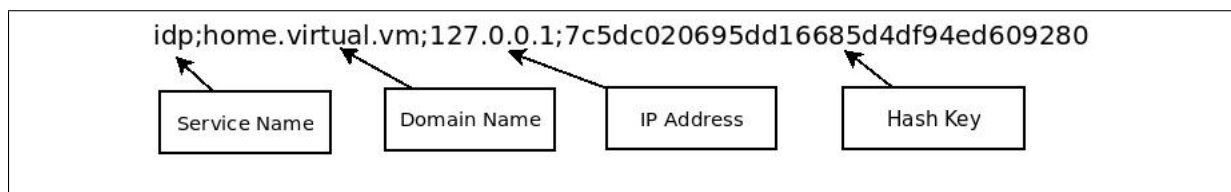


Figure 5.18: Sample Shared Key between Identity Provider Simulation and Kerberos Server Simulation.

### Requesting Service Ticket

The ticket granting ticket, service ticket and shared key are simulations of real world TGT, ST and shared key. The real world web browsers (e.g. Mozilla Firefox, Google Chrome and Microsoft Internet Explore) support real world ticket granting tickets; they do not support these simulations. This issue is addressed by designing a file uploading system, it enables the end-users to manually upload the simulation of ticket granting ticket (figure 5.19) onto the identity provider simulation. In the real world, web-browsers should automatically access the Kerberos ticket granting ticket.

The screenshot shows a web interface for uploading a Kerberos TGT. It features a label 'Kerberos TGT:', followed by a 'Choose File' button, the text 'No file chosen', and a 'Submit' button.

Figure 5.19: Kerberos Ticket Granting Ticket Portal Design.

When an end-user uploads a ticket granting ticket (with the ticket uploading system), the identity provider simulation passes the content of TGT to the Kerberos server simulation. As described in section 5.9.4, the Kerberos server simulation performs verifi-

cations on the TGT. If the verification was successful, the Kerberos server simulation uses hash function to create a service ticket. The service ticket is sent to the identity provider simulation. The identity provider simulation verifies the service ticket by comparing its content (the hash key string) with the hash key in the shared key. If the service key matches the hash key in the shared key, the end-user is authenticated; if the service key does not match the hash key in the shared key, the end-user is not authenticated.

### Create Identity Assertion

In order for end-users to access computer services in domain Beta, identity provider simulation needs to provide identity assertions for these end-users. The established federation between domain Alpha and Beta allows end-users from domain Alpha to access computer services in domain Beta. Once the end-user is successfully authenticated, the identity provider simulation will provide identity assertions to indicate that the end-users are from domain Alpha. Only with this identity assertion, the end-users can access computer service in domain Beta.

The simulation of identity assertion message includes the following variables:

- Username - the username of the end-user.
- Role - the role of the end-user.
- Domain name - the domain name (i.e. home.virtual.vm)
- Authentication Result - This variable is created when an end-user has been successfully authenticated by the identity provider simulation. The value is “true”.

Upon successful authentication, these four variables will be stored as session variables in the web session. For web-based services, web sessions are used to maintain communication between end-users and web-based services. When an end-user navigates the identity provider simulation, a web session starts. The web session is maintained by the web-browser. When an end-user closes its web-browser, the web session ends. In a web session, a session variable can be accessed by multiple web pages. In this design, an

end-user identity assertion message is saved as session variables in the web session. When the end-user navigated to the service provider simulation's website, the service provider simulation can access the end-user's identity assertion variables in the web session.

### 5.9.7 SAML based FSSO Simulation Design (Service Provider)

A simulation of a web-based service (i.e. service provider) is presented in this section. The goal of this simulation is to display the authentication detail of the end-users. It will not provide any actual web services.

Upon receiving an identity assertion message, the service provider simulation will verify the identity assertion message. If the identity assertion indicates that the end-user is from domain Alpha, the federation policies requires the end-user's role to be mapped into a different role in domain Beta (described in section 5.9.3) .

#### Access Identity Assertion

Section 5.9.6 shows that the end-users' identity assertions contain four variables, "user-name", "domain name", "role" and "authentication result". They are stored as session variables in the web sessions, service provider simulation can access them as long as the web sessions exist. The variable – "authentication result" – is a boolean variable (i.e has the value "true" or "false"). It indicates whether an end-user has been authenticated or not. An authenticated end-user will have the "true" value in its "authentication result". It indicates that the end-user is from domain Alpha. Service provider simulation looks for the variable "authentication result". If the "authentication result" is "true" in an end-user's identity assertion message, it means this end-user has been successfully authenticated by domain Alpha (i.e. this end-user is from domain Alpha). The end-user is in titled (i.e. according to the federation policies) to access web-based service in domain Beta.

## Mapping the Roles of End-users

As described in section 5.9.3, domain Alpha assigns roles such as “student” and “lecture” to its end-users, while domain Beta assigns roles such as “power” and “guest” to its end-users. These roles are not interoperable between these two domains. The roles from domain Alpha can not be used for authorisation in domain Beta.

This research design a role mapping model that bases on the federation policies. Its goal is to demonstrate the changes of end-user’s credential information (i.e. the change of role), it does not have any effect on the over all system design. The federation policies states that the role “lecture” in domain Alpha is mapped into role “power” in domain Beta; the role “student” in domain Alpha is mapped in role “guest” in domain Beta. Before any end-user accesses web-based serviced in domain Beta, its role will be mapped into it’s correspondent role in domain Beta.

### 5.9.8 Middle-ware System Design

The middle-ware system expand upon the approaches of Kerberos Secure Sharing (Grid-wise Tech, 2010) and SAML-AAI/Kerberos (Papez, 2009). It is designed to use the established federated single sign-on policies and support federated single sign-on for non-web based services. The differences between this approach and the others are the following:

- It aims to incorporate the existing Kerberos single sign-on system (i.e. little modification to existing authentication infrastructure).
- It aims to minimums the modification to existing computer services/applications.

This middle-ware system contains two components, identity provider ( $M_\alpha$ ) and service provider ( $M_\beta$ ). According to the system requirements, there are two domains, Alpha and Beta. Domain Alpha provides identities for the end-users and domain Beta provides services. According to the federation’s policies, end-users of domain Alpha can access computer services in domain Beta.  $M_\alpha$  resides in domain Alpha; it authenticates the end-users and provides identity assertions.  $M_\beta$  resides in domain Beta; it receives

identity assertions and provides non-web based services to the end-users. Section 5.9.8 describes the design of the middle-ware system's identity provider; section 5.9.8 describes the design of the middle-ware system's service provider.

### Design of the Middle-ware System's Identity Provider

This section describes the design of the middle-ware system's identity provider ( $M_\alpha$ ).  $M_\alpha$  relies on Kerberos server simulation for authentication. Kerberos server simulation provides ticket granting ticket and service ticket to the end-users; it also provide shared key to  $M_\alpha$ .  $M_\alpha$  verifies the service ticket of the end-users and provides identity assertions. It has the following functions:

- Requesting shared key from Kerberos server simulation.
- Verifying the end-users' service tickets.
- Creating and sending the identity assertion messages.
- Receiving the service tickets from domain Beta and passing them to the end-users.

#### Requesting Shared Key

To request a shared key, the  $M_\alpha$  needs to be registered in Kerberos server simulation's service database. As described in section 5.9.4, a service database stores services data (e.g. service name, domain name and IP address).  $M_\alpha$  is stored under the service name "saml-aai-kerberos", domain name "home.virtual.vm" and sample IP address "127.0.0.1"; these data are stored in XML format (figure 5.20).

```
<?xml version="1.0" encoding="iso-8859-1"?>
<services>
  <service>
    <serviceName>saml-aai-kerberos</serviceName>
    <domain>home.virtual.vm</domain>
    <ipAddress>127.0.0.1</ipAddress>
  </service>
</services>
```

Figure 5.20: Sample Identity Provider Entry in the Service Database.

$M_\alpha$  uses network socket to set up a network connection to Kerberos server simulation. It sends a request for a shared key to the Kerberos server simulation. Upon receiving the request, the Kerberos server simulation verifies that  $M_\alpha$  has been registered in the service database. The verification function was described in section 5.9.4. Since  $M_\alpha$  has been manually registered, the Kerberos server simulation creates a shared key. The shared key consists of  $M_\alpha$ 's service name, domain name and IP address and a hash key (figure 5.21). The service name, domain name, IP address are strings; they are merged together to a single "source string" and separated by semicolons. The hash key is created by putting the "source string" through SHA-2 hash function.

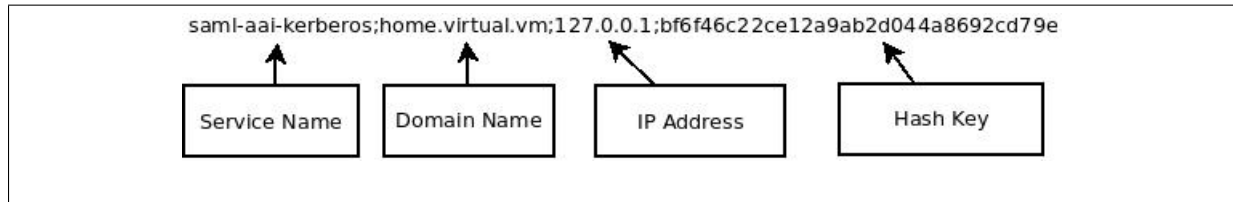


Figure 5.21: Sample Shared Key of Middle-ware system's Identity Provider.

### Verifying Service Ticket

Before creating identity assertions for the end-users, the middle-ware system's identity provider ( $M_\alpha$ ) needs to verify the service ticket of the end-users. The verification process is the same as SSH server simulation (section 5.9.5) and SAML based identity provider (section 5.9.6). An end-user sends its username and service ticket to  $M_\alpha$ .  $M_\alpha$  compares the hash key in the share key with the service ticket.

### Creating and Sending Identity Assertion

If the service ticket was successful verified,  $M_\alpha$  will send a response to the end-user. This response indicates that the end-user's service ticket has been verified. Upon receiving the response, the end-user sends its username, role and domain name to  $M_\alpha$ . These data are merged into a single string and they are individually separated by semicolon within the string (figure 5.22). This string is used as the identity assertion message.

$M_\alpha$  connects to the middle-ware system's service provider ( $M_\beta$ ) in domain Beta. The connection is created with network sockets. The identity assertion message is sent to  $M_\beta$  along with a request for service ticket.



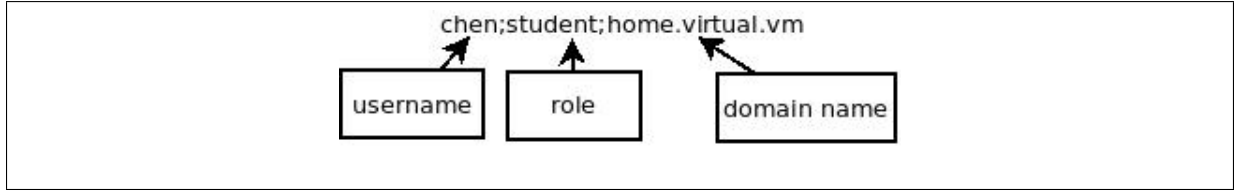


Figure 5.22: Sample Identity Assertion.

### Receiving and Passing the service tickets

Upon receiving the identity assertion message,  $M_\beta$  replies with a service ticket. This service ticket is created by the Kerberos server simulation in domain Beta. It allows end-users to access non-web based services in domain Beta. The detail processes of creating service ticket are described in section 5.9.8.  $M_\alpha$  receives the service ticket and sends it to the end-user.

### Design of the Middle-ware System's Service Provider

This section describes the design of the middle-ware system's service provider ( $M_\beta$ ). The main goals of  $M_\beta$  are requesting ticket granting ticket and service ticket on behalf of the end-users from domain Alpha. The functions of  $M_\beta$  are the following:

- Requesting ticket granting ticket on behalf of the end-user.
- Requesting service ticket on behalf of the end-user.

### Requesting Ticket Granting Tickets

The middle-ware system's service provider uses network sockets for network connection. It listens on a specific network port (port 1234). Once  $M_\alpha$  sends a connection request to  $M_\beta$  on port 1234,  $M_\beta$  will accept the connection request; this creates a network session between  $M_\alpha$  and  $M_\beta$ .

Through the established connection,  $M_\alpha$  sends end-users' identity assertion messages and request for service ticket to  $M_\beta$ . The identity assertion indicates that the correspondent end-user has been authenticated by  $M_\alpha$ , and the end-users are from domain Alpha. According to the federation's policies, end-users from domain Alpha can access services in domain Beta; therefore,  $M_\beta$  accepts the identity assertion from  $M_\alpha$ .

Section 5.9.2 described the process of identity provisioning; the end-users' identities were manually provisioned to the end-user database ( $D_\beta$ ) in domain Beta. The end-users' usernames and domain names stay the same after the identity provisioning; the end-users' password and role are changed after the identity provisioning. The end-users' password has changed from "123" to "321" in  $D_\beta$ , and the end-users' permission has been changed according to the federation's policies. The end-user, who has the role "student" in domain Alpha, has the role "guest" in domain Beta; the end-user, who has the role "lecture" in domain Alpha, has the role "power" in domain Beta.

One of the functions of  $M_\beta$  is requesting ticket granting ticket on behalf of the end-users. Once  $M_\beta$  accepts the identity assertion of an end-user, it extracts the username and domain name from the identity assertion message. It then searches for the correspondent identity in the end-user database in domain Beta. Since the identity of the end-user has been manually provisioned in the database,  $M_\beta$  should find the credential of the end-user in the database. It extracts the username and password of the end-user and sends it to the Kerberos server simulation ( $K_\beta$ ) in domain Beta, it also sends the request of ticket granting ticket to  $K_\beta$ .

Since  $K_\beta$  and  $M_\beta$  uses the same end-user database in domain Beta. The end-user should be authenticated successfully,  $K_\beta$  should create a ticket granting ticket (figure 5.23) for the end-user and send it to  $M_\beta$ . The ticket granting ticket is kept by  $M_\beta$ .



Figure 5.23: Sample Ticket Granting Ticket of Domain Beta.

### Requesting Service Ticket

The other function of  $M_\beta$  is requesting service ticket on behalf of the end-user. After requesting ticket granting ticket (TGT),  $M_\beta$  sends the TGT and request for service ticket to  $K_\beta$ . The service ticket is used to access non-web based service (in this case, SSH server simulation) in domain Beta.  $K_\beta$

$K_\beta$  verifies the TGT and create a service ticket. The detailed processes were described in section 5.9.4. It then returns the service ticket to  $M_\beta$ .  $M_\beta$  receives the service ticket and sends it to  $M_\alpha$ .

## 5.10 Summary

This chapter describes the design of the new simplified federated single sign-on system. It includes the designs of a novel middle-ware system, authentication infrastructures (based on Kerberos), web-based federated single sign-on infrastructure (based on SAML) and testing environment (with web-based and non-web based services/applications). The design of a novel middle-ware system is the main contribution of this research. It delivers federated single sign-on for non-web based services/applications. It is designed to minimise the deployment issues by incorporating (instead of modifying) network domains' existing authentication infrastructures and to minimise the modifications to existing computer services/applications. The design shows that the middle-ware system can incorporate existing authentication infrastructures, however, it can not eliminate the additional modifications to the existing computer services/applications. Computer services/applications requires additional functions to distinguish between local network single sign-on and cross domain single sign-on (i.e. federated single sign-on).

# Chapter 6

## Implementation of the New Simplified FSSO System

### 6.1 Introduction

The development of the new simplified federated single sign-on system includes the proof of concept implementation of the system design. This chapter presents the implementation of the design environment (i.e. network domains), implementation of the simulations (e.g. Kerberos single sign-on system, SAML based federated single sign-on system, web-based services/applications and non-web based services/applications) and the implementation of the middle-ware system.

Section 6.2 presents the technical overview. It includes the development environment (e.g. computer software and programming language) and system configuration (e.g. hardware and software configuration). Section 6.3 discusses the user manual of the implemented system. In the end, section 6.4 summaries the chapter.

## 6.2 Technical Overview

### 6.2.1 Development Environment and Programming Languages

The new simplified federated single sign-on system was developed on the Ubuntu 10.04.4 LTS Server and Xubuntu 12.04 LTS platform using programming languages, Python, PHP and AJAX. Python 2.6 was used to develop the simulation of Kerberos single sign-on, the simulation of non-web based service and the middle-ware system. PHP 5.3.10 and AJAX were used to develop the simulation of SAML based federated single sign-on system and non-web based service.

The simulations of Kerberos single sign-on system and non-web based applications involve tasks such as communications between multiple applications. It requires the source code to be clean and easy to debug. This reason promoted the choice of Python language. Python promotes clean and easy to read code style, it's suitable for programming and debugging complex applications. The simulation of SAML based single sign-on and web-based application are small but dynamic web applications. These reasons promoted the used of PHP and AJAX. PHP is a server-side HTML embedded scripting language, it is suitable to create small scale web applications. AJAX creates asynchronous web applications, it can change contents on a web page without refreshing the web browser.

### 6.2.2 System Configuration

The new simplified federated single sign-on system was developed on Sun Ultra 40M2 Workstation and HP ProBook 6550b. These machines were issued by Ubiquitous Computing Research Group (UCRG) in School of Computing. Sun Ultra 40M4 Workstation is used as hosting servers for virtual machines. HP ProBook 6550b is a laptop machine that is hosting a desktop Linux operating system. The detail information on hardware configurations are presented in appendix A.1.

Sun Ultra 40M2 Workstation is used as hosting server for virtual machines. Two virtual machines are created with XenServer: a DNS server that manages the domain

name and a virtual server acts as domain Beta (with domain name `foreign.virtual.vm`). HP Probook 6550b is installed with Xubuntu Desktop 12.04 LTS 64bit; it acts as domain Alpha (with domain name `home.virtual.vm`). The Dell Optiplex GX260 is installed with Windows XP SP2 and XenCenter 6.0. It is used as graphic interface management software for XenServer. The detail information on software configuration are presented in appendix A.2.

## 6.3 The Simplified Federated Single Sign-On System User Manual

### 6.3.1 Kerberos Single Sign-on Server Simulation

This section presents the simulation of Kerberos single sign-on server. Since Kerberos server simulation is designed for local domain single sign-on, only domain Alpha is used in this demonstration.

Three python scripts are used in this demonstration:

- `idp_kdc_sim.py` is the python script that simulations the Kerberos server in domain Alpha,
- `idp_ssh_client_sim.py` simulates the local Unix/Linux desktop in domain Alpha,
- `idp_ssh_server_sim.py` simulates the remote Unix/Linux desktop in domain Alpha.

#### Initiate Kerberos Single Sign-on Server Simulation

To start the Kerberos single sign-on server simulation, the Test Administrator executes the following python script:

```
~$ ./idp_kdc_sim.py
```

This command executes the python script and starts the Kerberos server simulation (figure 6.1). The simulation has the IP address, 192.168.1.59, and it listens on the specified port, 50000.

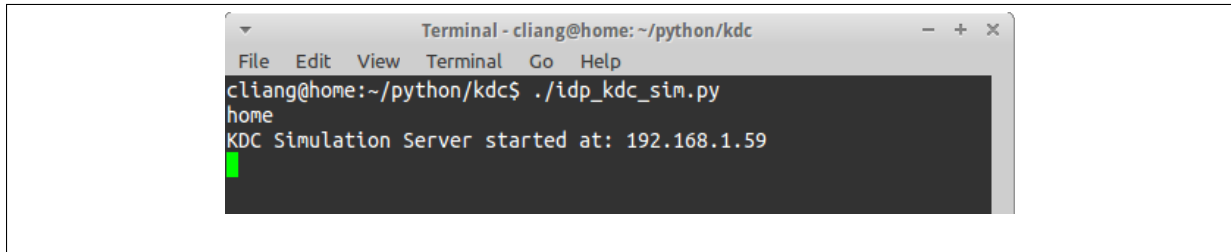


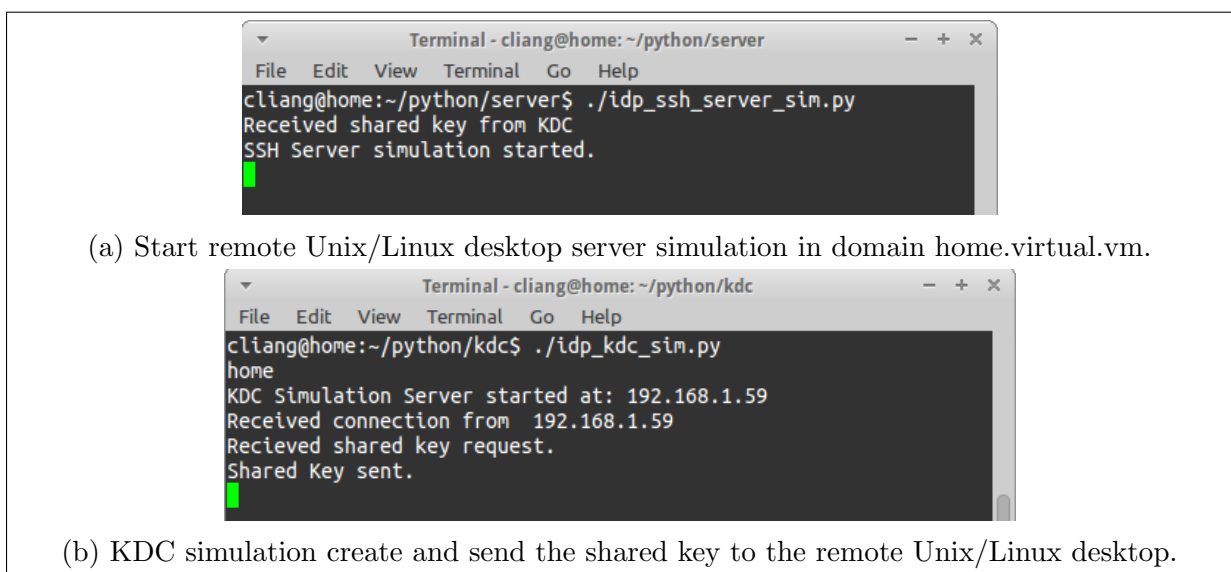
Figure 6.1: Start KDC Simulation in domain home.virtual.vm.

### Initiate Remote Unix/Linux Desktop Simulation

To start the remote Unix/Linux desktop simulation, the Test Administrator executes the following python script:

```
~$ ./idp_ssh_server_sim.py
```

This simulation of remote Unix/Linux desktop connects to the Kerberos server simulation and request for a shared key. Kerberos server simulation accepts the connection and returns the shared key (figure 6.2)



(a) Start remote Unix/Linux desktop server simulation in domain home.virtual.vm.

(b) KDC simulation create and send the shared key to the remote Unix/Linux desktop.

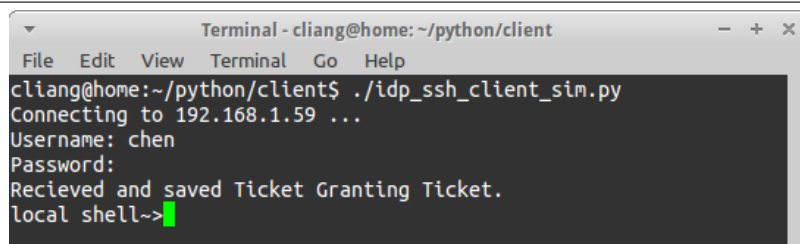
Figure 6.2: Remote Unix/Linux desktop simulation requests shared key from KDC simulation.

## Initiate Local Unix/Linux Desktop Simulation

To start the local Unix/Linux desktop simulation, the end-user executes the following python script:

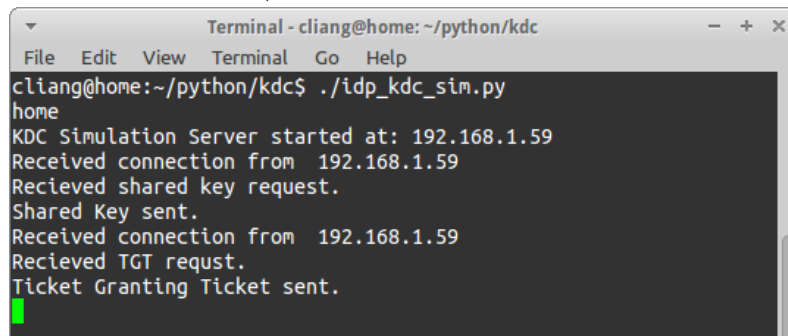
```
~$ ./idp_ssh_client_sim.py
```

The desktop simulation will promote an authentication prompt. The end-user enter its username/password, which is sent to the Kerberos server simulation. The Kerberos server simulation verifies the credential and returns a ticket granting ticket to the end-user (figure 6.3).



```
Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
cliang@home:~/python/client$ ./idp_ssh_client_sim.py
Connecting to 192.168.1.59 ...
Username: chen
Password:
Recieved and saved Ticket Granting Ticket.
local shell~>
```

(a) End-user login the local Unix/Linux desktop simulation in domain home.virtual.vm.



```
Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp_kdc_sim.py
home
KDC Simulation Server started at: 192.168.1.59
Received connection from 192.168.1.59
Recieved shared key request.
Shared Key sent.
Received connection from 192.168.1.59
Recieved TGT request.
Ticket Granting Ticket sent.
```

(b) KDC simulation authenticates the end-user and creates ticket granting ticket.

Figure 6.3: End-user login the local Unix/Linux desktop and receives ticket granting ticket.

## Demonstrate the Local Unix/Linux Desktop Simulation

As described in chapter 5, the local Unix/Linux desktop simulation supports the following commands:

- “ls” - Listing the contents (files and directories) in the current directory.



- “touch” - Create a text file.
- “hostname” - Display the hostname of the current Unix/Linux desktop simulation.
- “ifconfig” - Display the IP address of the current Unix/Linux desktop simulation.
- “klist” - Listing Kerberos ticket granting tickets on the current Unix/Linux desktop simulation.
- “ssh” - Secure shell protocol simulation.
- “exit” - Exiting the system. Termination of simulations will not remove any saved files.

This section demonstrates the following commands: “klist”, “hostname”, “ifconfig”, “ls”.

After login the local Unix/Linux desktop, the end-user executes the listed commands to demonstrate the simulation (figure 6.4). Command “hostname” returns the result “home”; command “ifconfig” returns the IP address “192.168.1.59”. Command “klist” and “ls” returns their respective results.

### **Access the Remote Unix/Linux Desktop Simulation**

To access the remote Unix/Linux desktop simulation, the end-user executes the following command:

```
local shell~>ssh remote.home.virtual.vm
```

The local Unix/Linux sends the end-user’s ticket granting ticket to the Kerberos server simulation along with the name of the service (i.e. remote Unix/Linux desktop). Kerberos server simulation verifies the ticket granting ticket and returns a service ticket. The service ticket is then sent to the remote Unix/Linux desktop by the local desktop. Remote Unix/Linux desktop verifies the service ticket and grants the end-user’s access (figure 6.5).

```

Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
cliang@home:~/python/client$ ./idp_ssh_client_sim.py
Connecting to home ...
Username: chen
Password:
Recieved and saved Ticket Granting Ticket.
local shell~>help
ls      ifconfig      hostname      exit      klist      ssh remote
.home.virtual.vm      ssh foreign.virtual.vm
local shell~>klist

Ticket Granting Ticket:
Username: chen
Password: 123
Role: student
Domain: home.virtual.vm
Ticket Create Date: 2012-06-21 14:05
Ticket Expire Date: 2012-06-22 14:05
Key: a35bc0e077718d06029902bcf9b92923

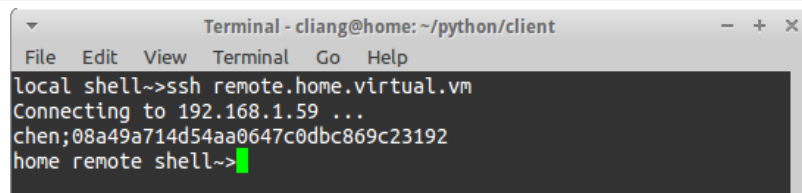
local shell~>ifconfig
192.168.1.59
local shell~>hostname
home
local shell~>ls
./Ticket.py
./idp_ssh_client_sim.py
./tgt.key
./idp_ssh_client_sim.py~
./Ticket.pyc
local shell~>

```

Figure 6.4: Command, “klist”, display the end-user’s ticket granting ticket; command, “hostname” displays the hostname of the current machine; command, “ifconfig” displays the local IP address of the current machine. Command, “ls”, display all the files in current directory.

### Demonstrate the Remote Unix/Linux Desktop Simulation

The end-user connects to the remote Unix/Linux desktop simulation and creates a test file 6.6. It lists all the files in the current directory on the simulation; it then creates a test file, “file1” in that directory. Another “ls” command shows that the file, “file1”, has been created on the simulation. The remote Unix/Linux desktop simulation displays correspondent message to the end-user’s commands.

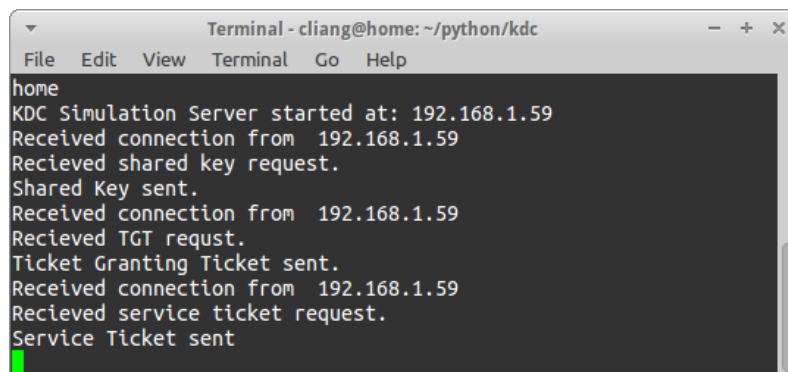


```

Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
local shell~>ssh remote.home.virtual.vm
Connecting to 192.168.1.59 ...
chen;08a49a714d54aa0647c0dbc869c23192
home remote shell~>

```

(a) End-user login a remote Unix/Linux desktop in domain home.virtual.vm with SSH connection.

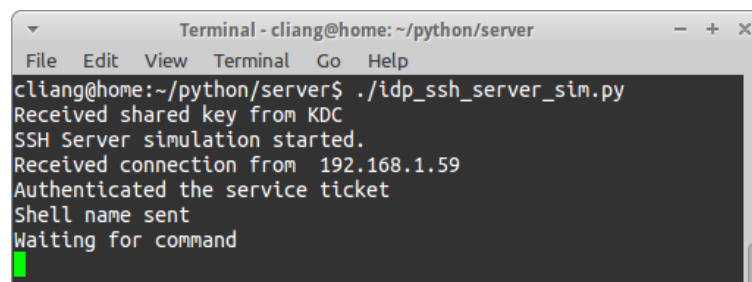


```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
home
KDC Simulation Server started at: 192.168.1.59
Received connection from 192.168.1.59
Received shared key request.
Shared Key sent.
Received connection from 192.168.1.59
Received TGT request.
Ticket Granting Ticket sent.
Received connection from 192.168.1.59
Received service ticket request.
Service Ticket sent

```

(b) KDC simulation verifies the TGT and sends service ticket to the end-user.



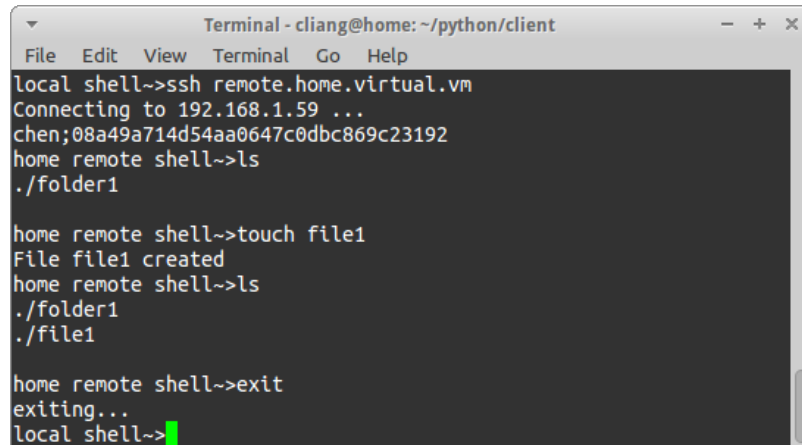
```

Terminal - cliang@home: ~/python/server
File Edit View Terminal Go Help
cliang@home:~/python/server$ ./idp_ssh_server_sim.py
Received shared key from KDC
SSH Server simulation started.
Received connection from 192.168.1.59
Authenticated the service ticket
Shell name sent
Waiting for command

```

(c) SSH server simulation verifies the service ticket and accepts the access from the end-user.

Figure 6.5: End-user login to the remote Unix/Linux desktop with secure shell.



```

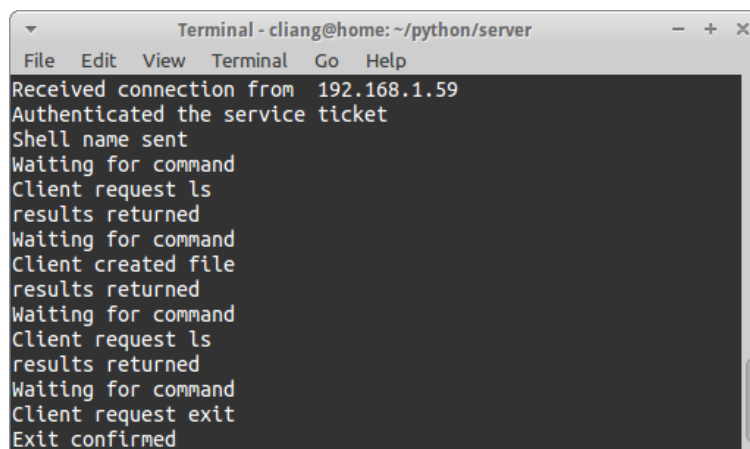
Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
local shell~>ssh remote.home.virtual.vm
Connecting to 192.168.1.59 ...
chen;08a49a714d54aa0647c0dbc869c23192
home remote shell~>ls
./folder1

home remote shell~>touch file1
File file1 created
home remote shell~>ls
./folder1
./file1

home remote shell~>exit
exiting...
local shell~>

```

(a) End-user creates files in the directory on remote Unix/Linux desktop in domain home.virtual.vm with SSH connection.



```

Terminal - cliang@home: ~/python/server
File Edit View Terminal Go Help
Received connection from 192.168.1.59
Authenticated the service ticket
Shell name sent
Waiting for command
Client request ls
results returned
Waiting for command
Client created file
results returned
Waiting for command
Client request ls
results returned
Waiting for command
Client request exit
Exit confirmed

```

(b) Remote Unix/Linux desktop simulation executes the commands.

Figure 6.6: End-user executes commands on the remote Unix/Linux desktop.

### 6.3.2 SAML based Federated Single Sign-On Simulation

This section presents the demonstration of SAML based federated single sign-on simulation. It involves domain Alpha and domain Beta. These applications are used in this demonstration:

- `sp_kdc.py` is the python script that simulates Kerberos single sign-on. It resides in domain Beta.
- `sp_ssh_server.py` is the python script that simulates remote Unix/Linux desktop in domain Beta.
- `sp.php`, `app.php` and `ip.php` are PHP web applications that simulate service provider in domain Beta.
- `idp.php`, `auth_success.php` and `auth_key.php` are PHP web applications that simulate identity provider in domain Alpha

#### Initiate the Kerberos Server Simulation

To start the Kerberos single sign-on server simulation, the Test Administrator executes the following python script:

```
~$ ./sp_kdc.py
```

This starts the Kerberos server simulation in domain Beta. It has the IP address 192.168.1.17. It listens on port 50000 for connection requests (figure 6.7).

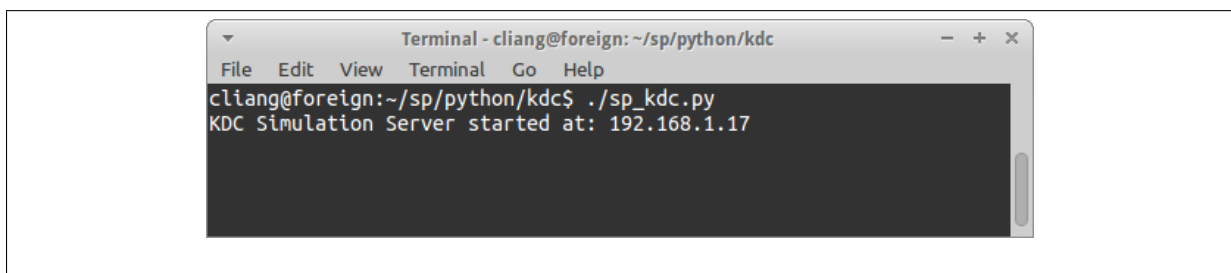


Figure 6.7: Start KDC in domain foreign.virtual.vm.

### Initiate the Remote Unix/Linux Desktop Simulation

To start the remote Unix/Linux desktop simulation, the Test Administrator executes the following python script:

```
~$ ./sp_ssh_server.py
```

The remote Unix/Linux desktop simulation connects to the Kerberos server simulation and requests a shared key. Kerberos server simulation accepts the connection request and returns a shared key to the remote desktop simulation (figure 6.8).

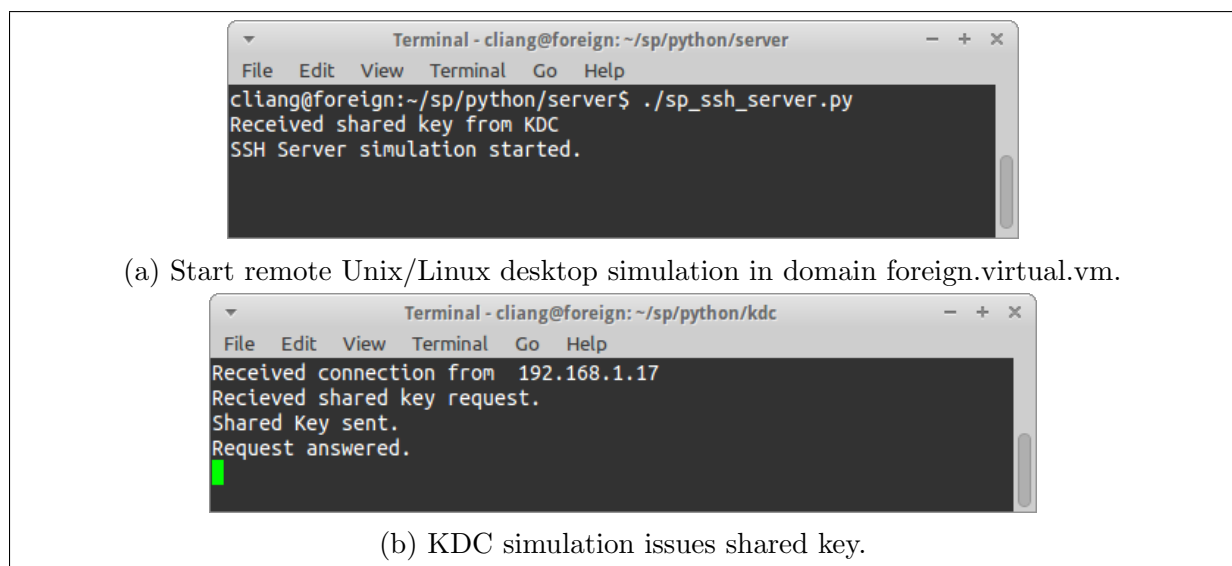


Figure 6.8: Remote Unix/Linux desktop simulation in domain foreign.virtual.vm.

### Demonstration the SAML-based FSSO Simulation

The end-user starts a web-browser and navigates to domain Beta's service provider's web-site. The service provider indicates that the end-user needs to be authenticated (figure 6.9). It also presents a list of domains that had set up federation with the service provider's domain.

The end-user chooses domain Alpha and the service provider re-directs the web browser to domain Alpha's identity provider. There are two authentication methods, username/password and Kerberos single sign-on. Since the end-user has been authenticated by the local Unix/Linux desktop in the last section, the end-user can use the ticket

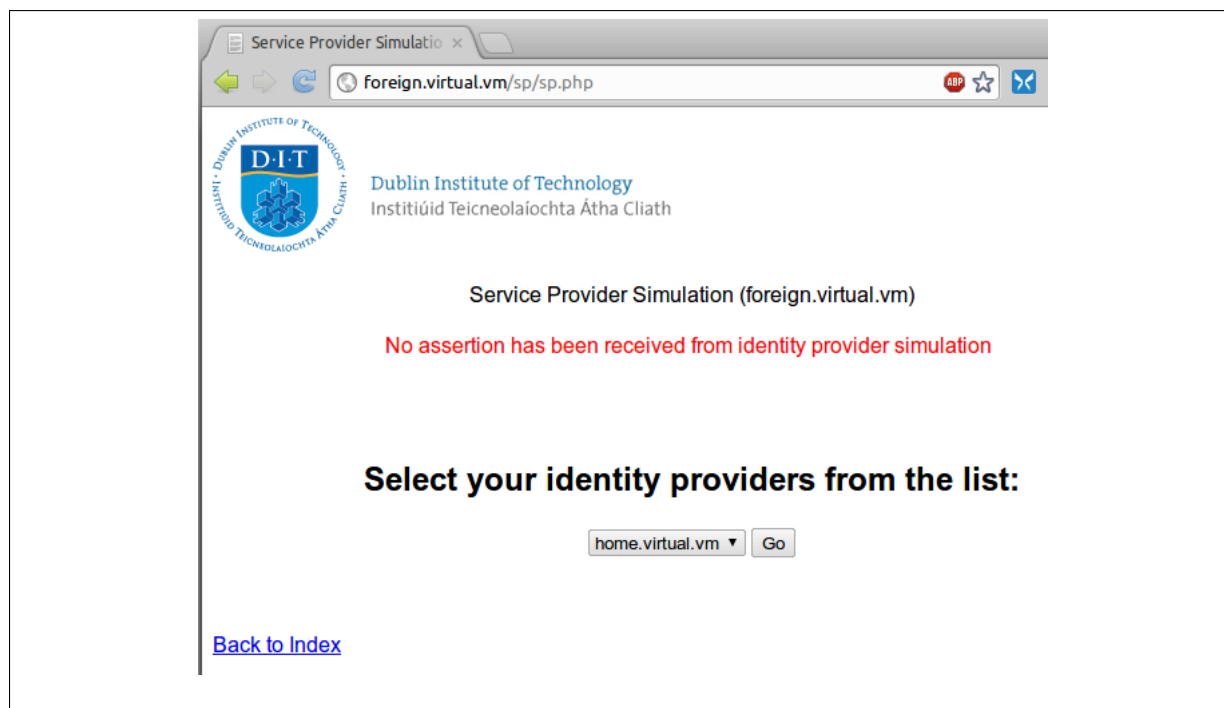


Figure 6.9: Access Web-based Service Provider in domain `foreign.virtual.vm`.

granting ticket for authentication. In this simulation, the end-user needs to upload its ticket granting ticket (figure 6.10).

When the identity provider verifies the end-user's identity (with Kerberos single sign-on), it displays the end-user's identity assertion message. The message includes the end-user's username, the issue date of the message, the end-user's domain and the end-user's role (figure 6.11).

The end-user accesses the service provider with the identity assertion message. The service provider in domain Beta verifies the identity assertion and requests a ticket granting ticket (TGT) and service ticket (ST) on behalf of the end-user. The TGT and ST are created by the Kerberos server simulation in domain Beta. The ST allows the end-user to access service in domain Beta.

A web-based ssh client allows the end-user to access the remote Unix/Linux desktop in a web-based manner. The service provider in domain Beta re-directed the end-user to the web-based ssh client along with the service ticket. The web-based ssh client sends the service ticket to the remote Unix/Linux desktop, once it is verified, the end-user is granted the access (figure 6.13). The end-user executes the commands "ls", "hostname"

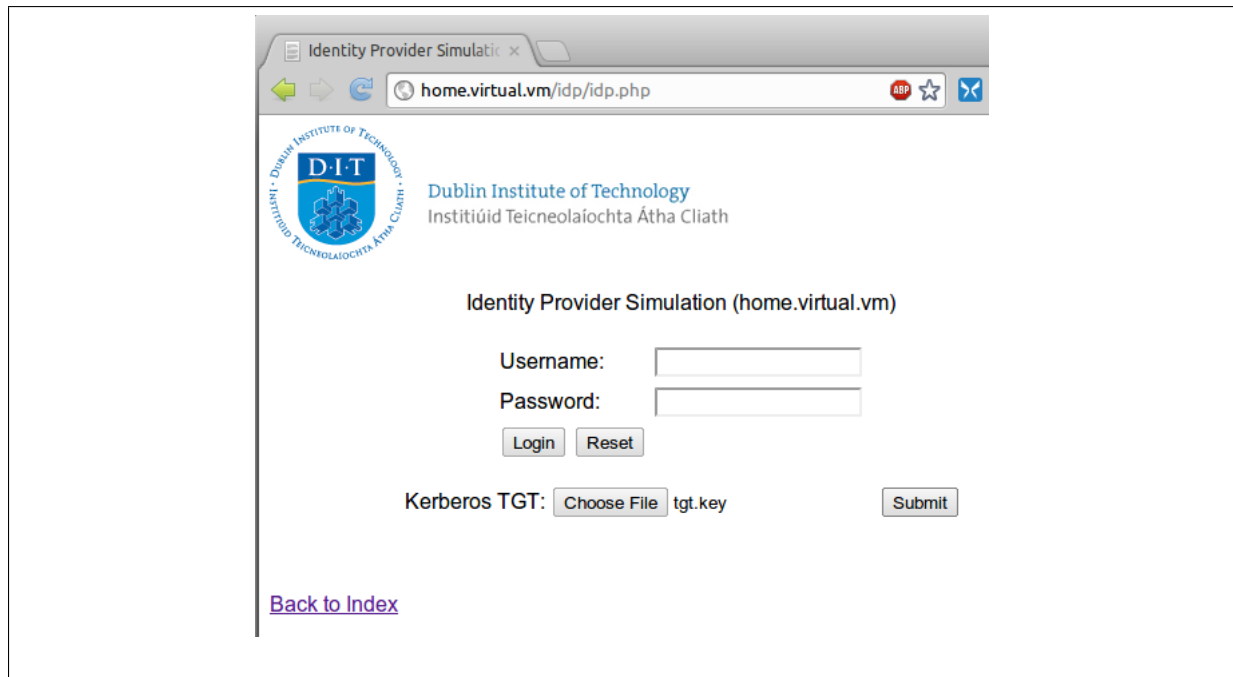


Figure 6.10: Use Kerberos Authentication Method in domain home.virtual.vm.

and “ifconfig” (figure 6.14). The results are as follows:

- “ls” displays all the files in the end-user’s home directory.
- “hostname” displays the hostname of the remote Unix/Linux desktop simulation.
- “ifconfig” displays the IP address of the remote Unix/Linux desktop simulation.



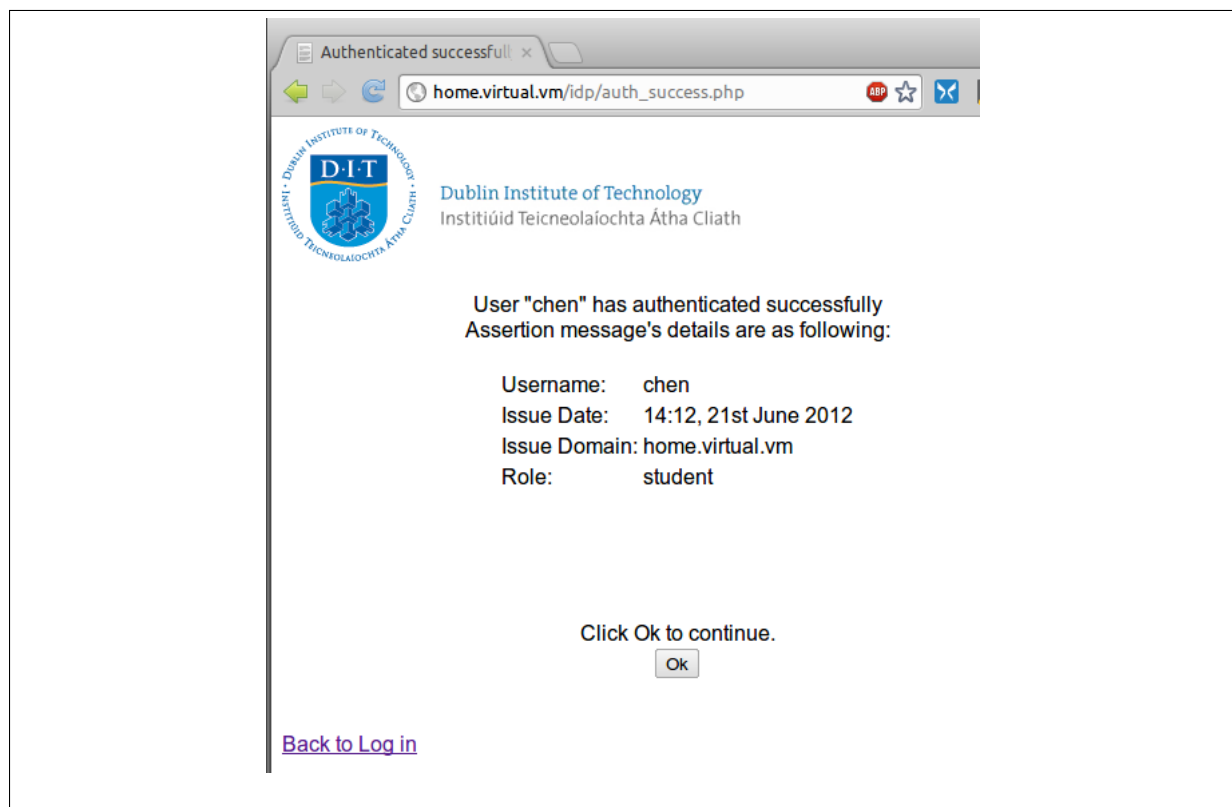


Figure 6.11: The end-user is successfully authenticated.

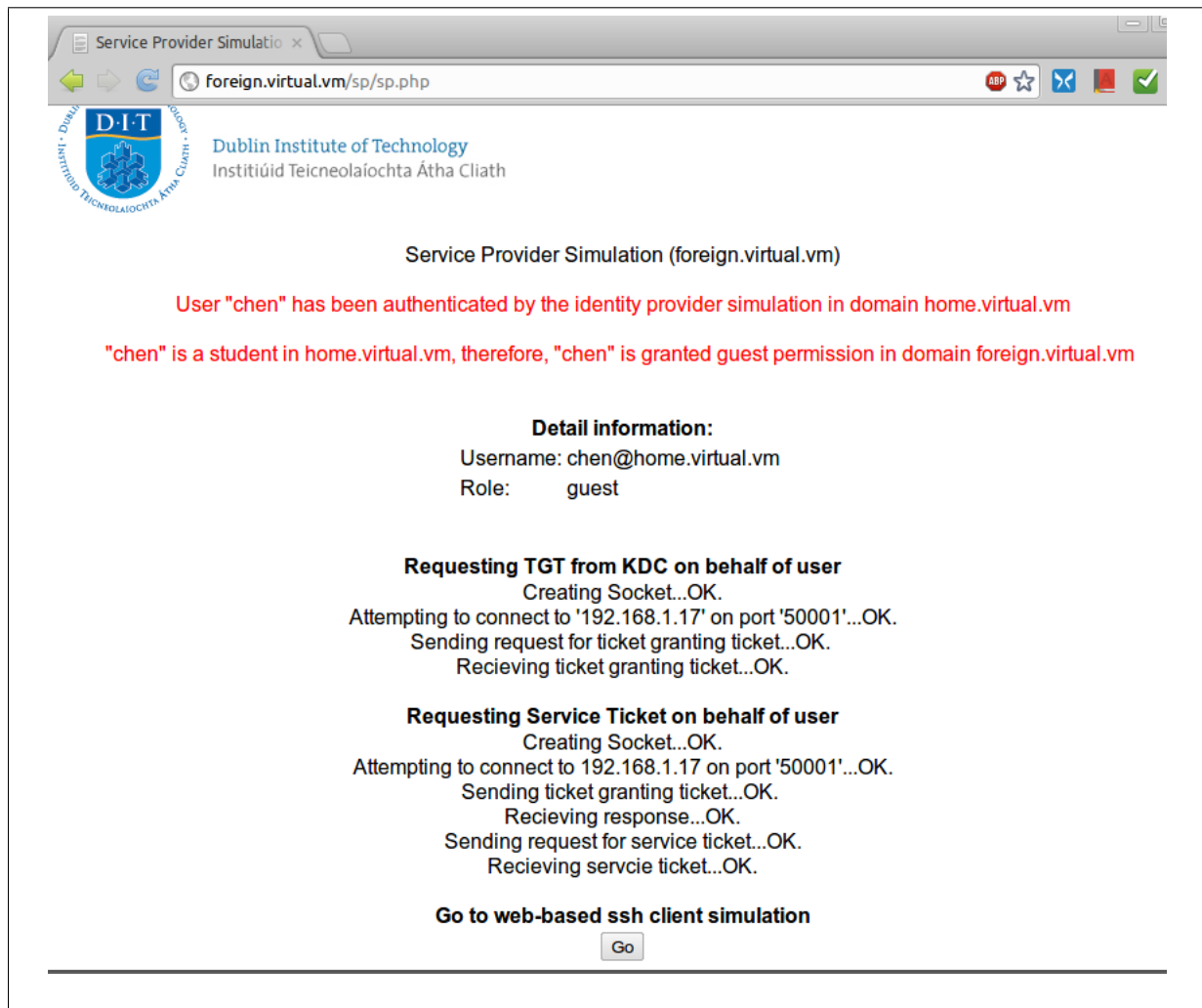


Figure 6.12: SAML Based Federated Single Sign-On Simulation in domain foreign.virtual.vm.

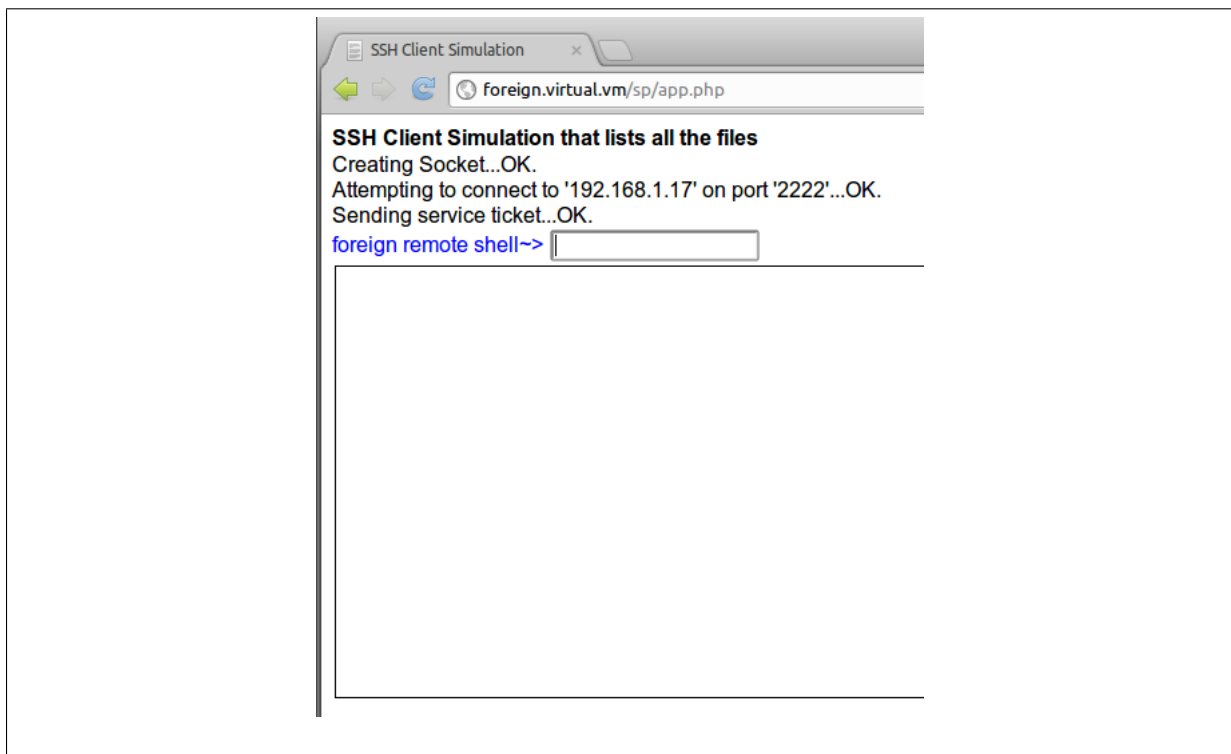
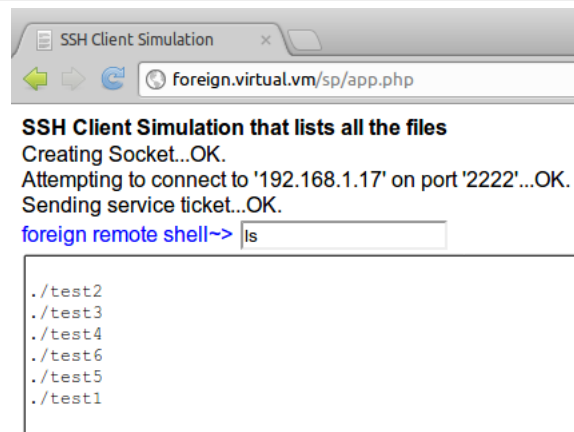
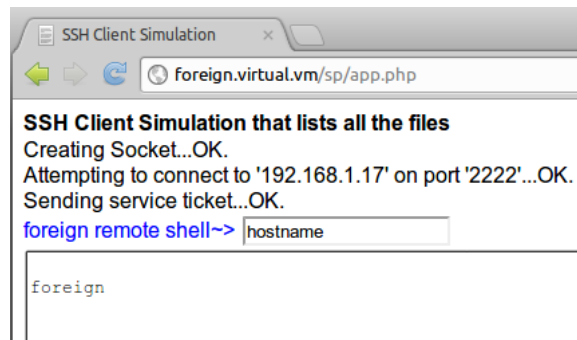


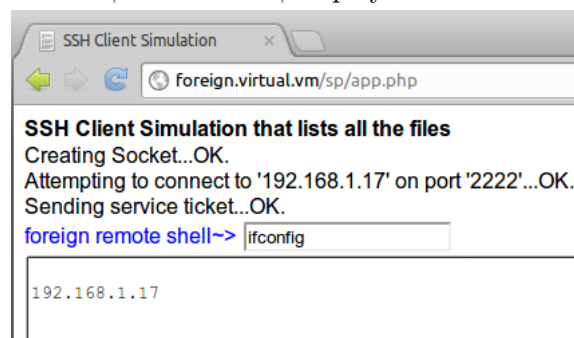
Figure 6.13: We-based interface for accessing remote Unix/Linux desktop simulation.



(a) Command, “ls”, lists all the files in the end-user’s home directory.



(b) Command, “hostname”, displays the local hostname .



(c) Command, “ifconfig”, displays the local IP address .

Figure 6.14: Executes sample Unix/Linux commands.

### 6.3.3 Accessing Non-web based Service in Another domain

This section demonstrates the middle-ware system, it aims to allow the end-user to access non-web based services with a single authentication session. It includes the following python scripts:

- `idp_ssh_client_sim.py` is a python script that simulate a local Unix/Linux desktop in domain Alpha.
- `idp_kdc_sim.py` is a python script that simulate a Kerberos server in domain Alpha.
- `idp-saml-aai-kerberos.py` is a python script that demonstrate the middle-ware system (identity provider end) in domain Alpha.
- `sp_ssh_server.py` is a python script that simulates a remote Unix/Linux desktop in domain Beta.
- `sp_kdc.py` is a python script that simulates a Kerberos server in domain Beta.
- `sp-saml-aai-kerberos.py` is a python script that demonstrate the middle-ware system (service provider end) in domain Beta.

#### Initiate the Kerberos server simulations

To start the Kerberos server simulations, the Test Administrator executes the following python scripts:

```
~$ ./idp_kdc_sim.py
~$ ./sp_kdc.py
```

The Kerberos server simulation in domain Alpha and domain Beta will listen for requests in their respective domains (figure 6.15).

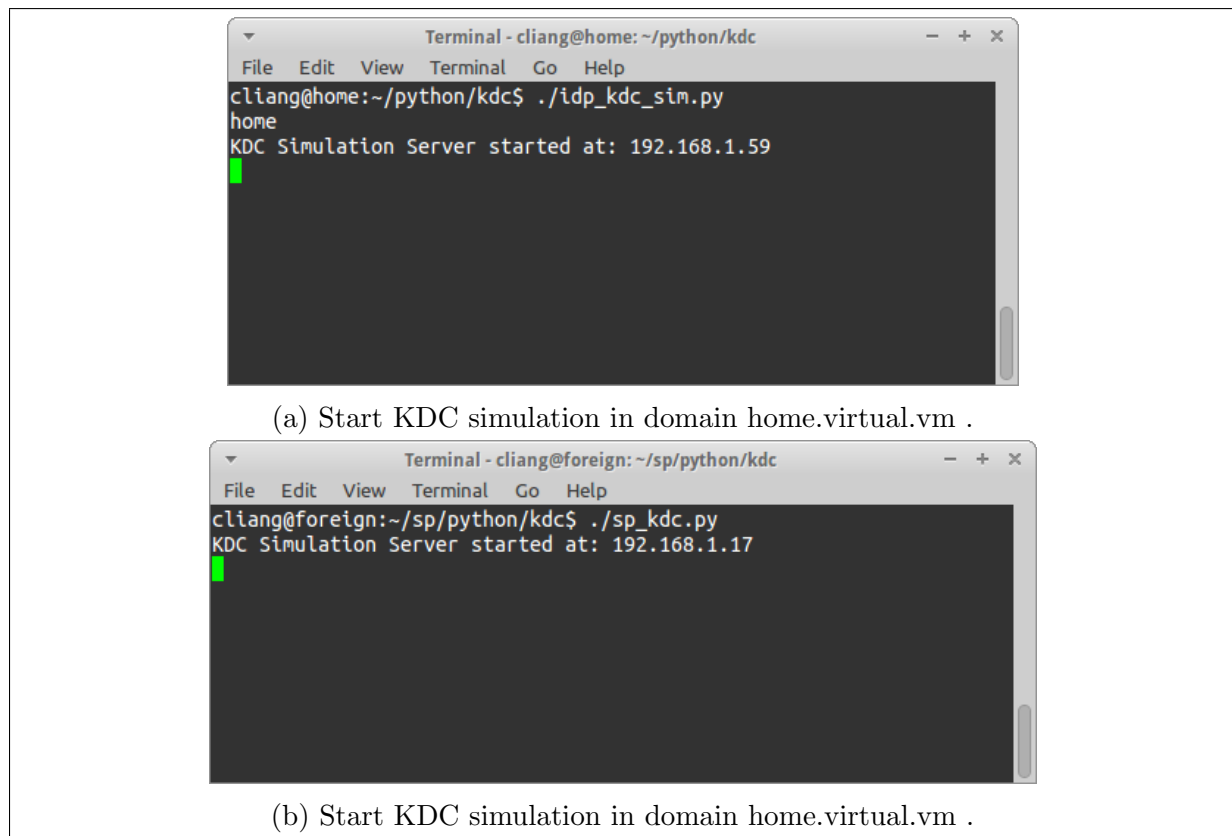


Figure 6.15: Start KDC simulation in domain home.virtual.vm and foreign.virtual.vm .

### Initiate the Middle-ware Systems in Domain Alpha

To start the Middle-ware System in domain Alpha, the Test Administrator executes the following python scripts:

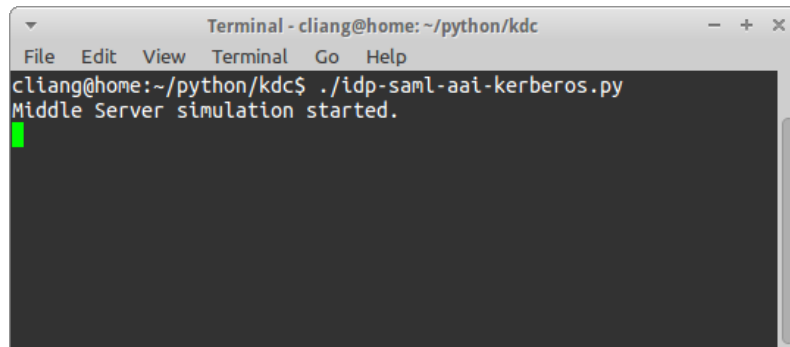
```
~$ ./idp-saml-aai-kerberos.py
```

The Middle-ware system in domain Alpha requests a shared key from the Kerberos server simulation. Kerberos server simulation issues a shared key. Then the middle-ware system will listen for requests in domain Alpha (figure 6.16).

### Initiate the Local Unix/Linux Desktop Simulation in Domain Alpha

To start the local Unix/Linux desktop simulation in domain Alpha, the Test Administrator executes the following python scripts:

```
~$ ./idp_ssh_client_sim.py
```

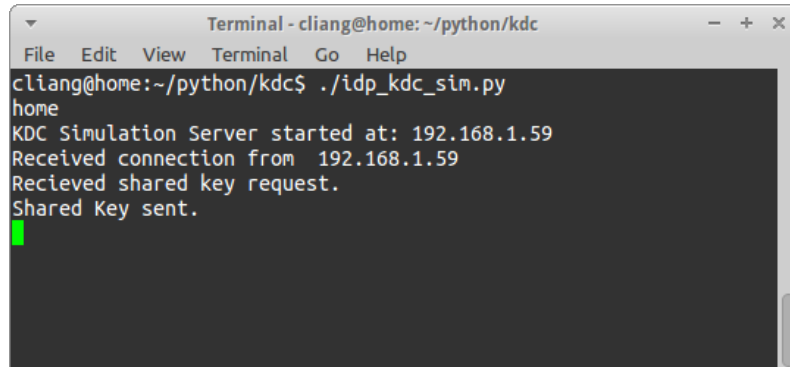


```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp-saml-aa-kerberos.py
Middle Server simulation started.

```

(a) Start middle-ware system (identity provider) in domain home.virtual.vm .



```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp_kdc_sim.py
home
KDC Simulation Server started at: 192.168.1.59
Received connection from 192.168.1.59
Recieved shared key request.
Shared Key sent.

```

(b) KDC simulation (in domain home.virtual.vm) issues shared key .

Figure 6.16: Start middle-ware system (identity provider) in domain home.virtual.vm .

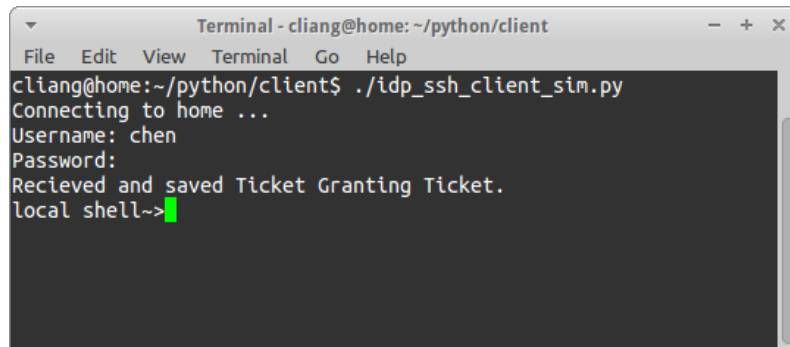
The end-user needs to enter its username and password. The pair of credential is sent to the Kerberos server simulation in domain Alpha. Upon successfully verifying the credentials, the Kerberos server simulation creates and sends the ticket granting ticket to the end-user's desktop simulation (figure 6.17).

### Initiate the Remote Unix/Linux Desktop in Domain Beta

To initiate the remote Unix/Linux desktop simulations in domain Beta, the Test Administrator executes the following python scripts:

```
~$ ./sp_ssh_server.py
```

The remote Unix/Linux desktop simulation connects to the Kerberos server simulation in domain Beta and requests a shared key. The Kerberos server simulation accepts the connection and returns a copy of the shared key (figure 6.18).

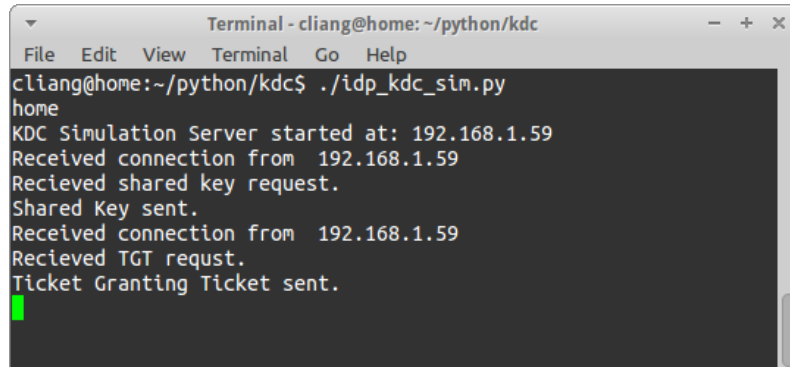


```

Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
cliang@home:~/python/client$ ./idp_ssh_client_sim.py
Connecting to home ...
Username: chen
Password:
Recieved and saved Ticket Granting Ticket.
local shell~>

```

(a) End-user login the Unix/Linux desktop simulation in domain home.virtual.vm .



```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp_kdc_sim.py
home
KDC Simulation Server started at: 192.168.1.59
Received connection from 192.168.1.59
Recieved shared key request.
Shared Key sent.
Received connection from 192.168.1.59
Recieved TGT request.
Ticket Granting Ticket sent.

```

(b) KDC simulation (in domain home.virtual.vm) authenticates the end-user and issues ticket granting ticket .

Figure 6.17: End-user login the Unix/Linux desktop simulation and receives a ticket granting ticket in home.virtual.vm .

### Initiate the Middle-ware system in domain Beta

To initiate the middle-ware system in domain Beta, the Test Administrator executes the following python scripts:

```
~$ ./sp-saml-aai-kerberos.py
```

The middle-ware system connects to the Kerberos server simulation in domain Beta and requests a shared key. Kerberos server simulation accepts the connection and returns a copy of the shared key (figure 6.19).

### Demonstrate the Middle-ware System

The end-user executes the command “ssh foreign.virtual.vm” in the local Unix/Linux desktop simulation (figure 6.20). The remote Unix/Linux desktop simulation requests



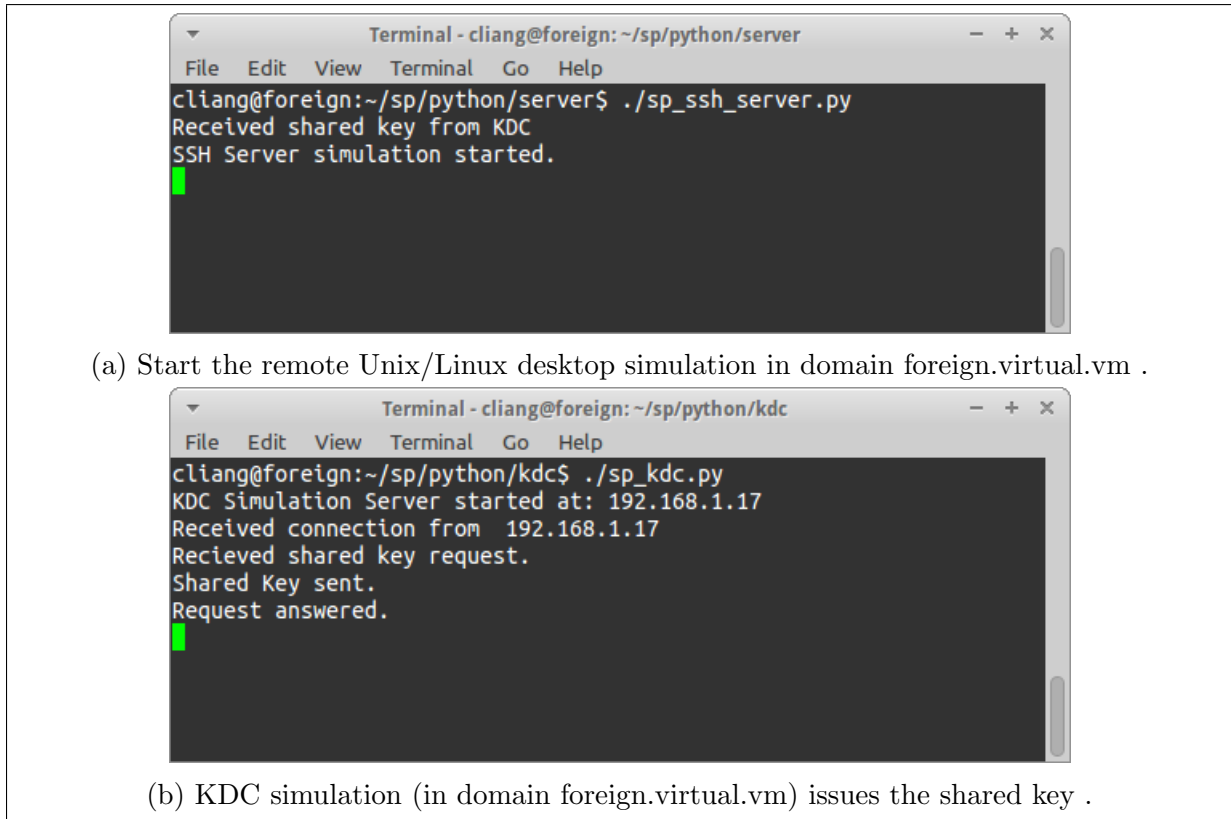


Figure 6.18: The remote Unix/Linux desktop simulation in domain foreign.virtual.vm receives a shared key .

authentication. This triggers the local Unix/Linux desktop simulation to request a service ticket for accessing the middle-ware system in domain Alpha.

The Kerberos server simulation in domain Alpha verifies the end-user's ticket granting ticket. It then sends a service ticket back to the end-user (figure 6.21).

The service ticket is then used to access the middle-ware system ( $M_\alpha$ ) in domain Alpha.  $M_\alpha$  verifies the service ticket and create an identity assertion for the end-user. It then sends the identity assertion to the middle-ware system ( $M_\beta$ ) in domain Beta (figure 6.22).

$M_\beta$  verifies the identity assertion. It then request a ticket granting ticket and service ticket on behalf of the end-user (figure 6.23).

The Kerberos server simulation in domain Beta creates the ticket granting ticket and service ticket. It then sent it back to the  $M_\beta$ .  $M_\beta$  passes the service ticket to  $M_\alpha$ .  $M_\beta$  then passes it to the end-user in domain Alpha.

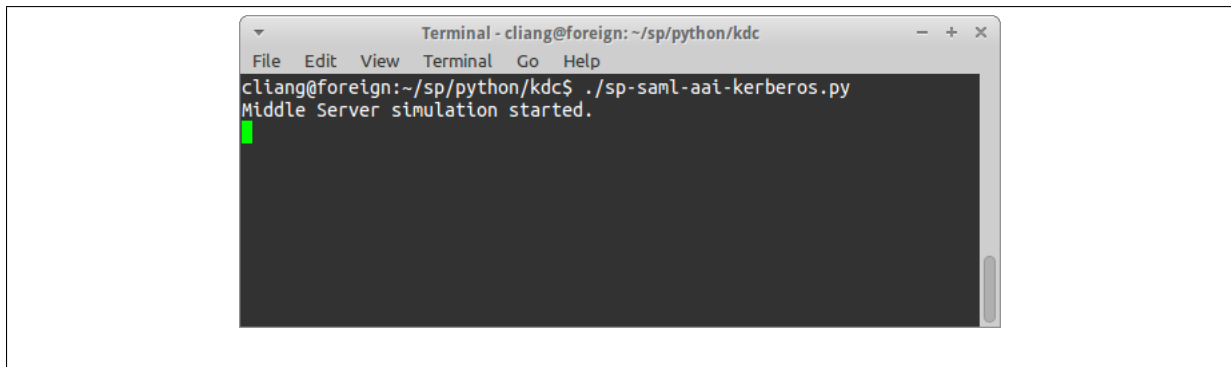


Figure 6.19: Start the middle-ware system (service provider) in domain foreign.virtual.vm .

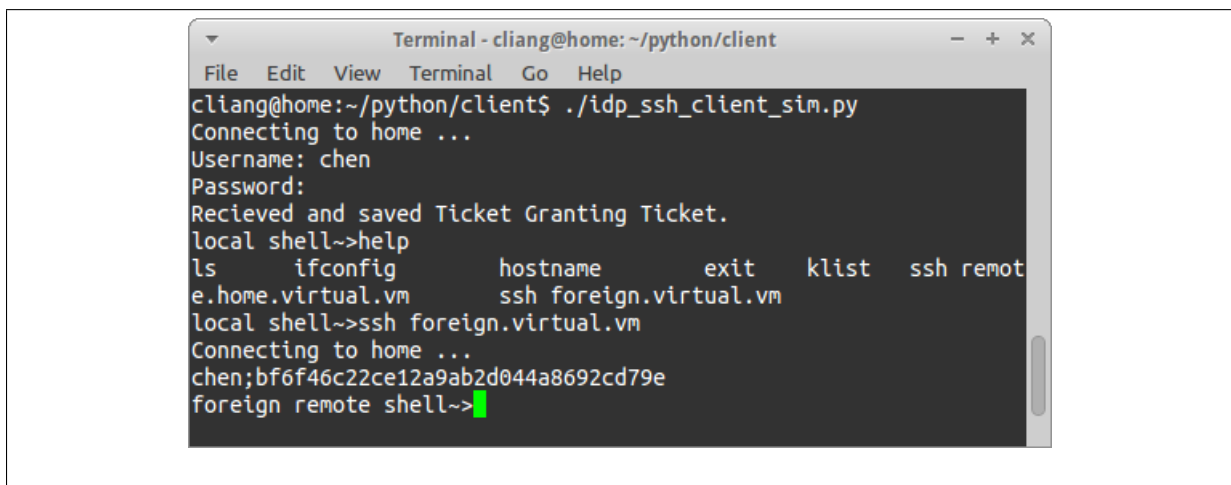
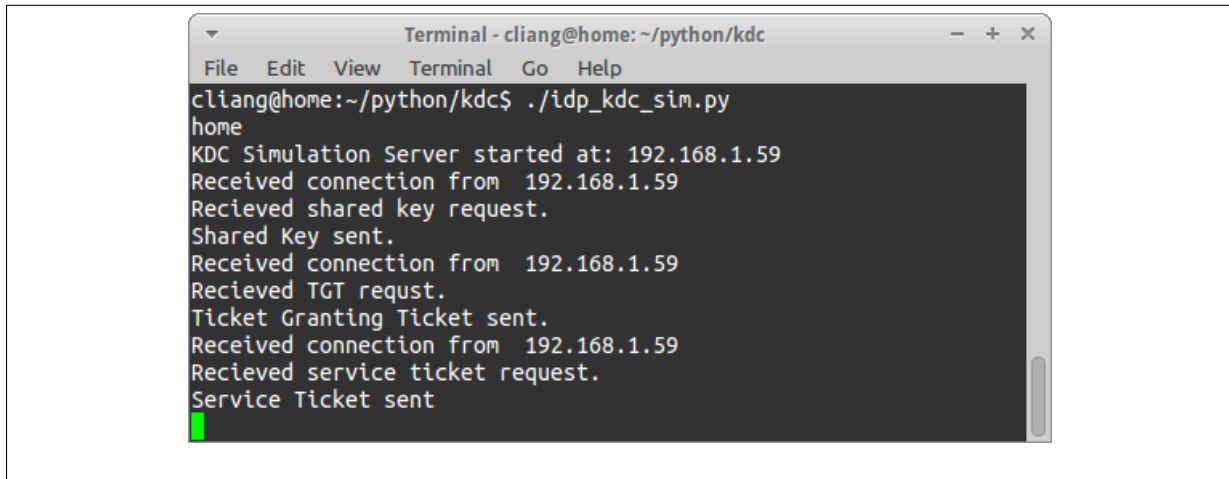


Figure 6.20: End-user attempts to access the Unix/Linux desktop in foreign.virtual.vm .

The end-user uses the service ticket to access the remote Unix/Linux desktop simulation in domain Beta. The remote desktop simulation verifies the end-user's service ticket, and it accepts the service request (figure 6.25).

Once the end-user gains access to the remote Unix/Linux desktop simulation in domain Beta, it performs commands "hostname", "ifconfig" and "ls". "hostname" displays the remote desktop simulation's hostname, "ifconfig" displays its IP address, and "ls" displays all the files in the current directory on the remote desktop simulation (figure 6.26).

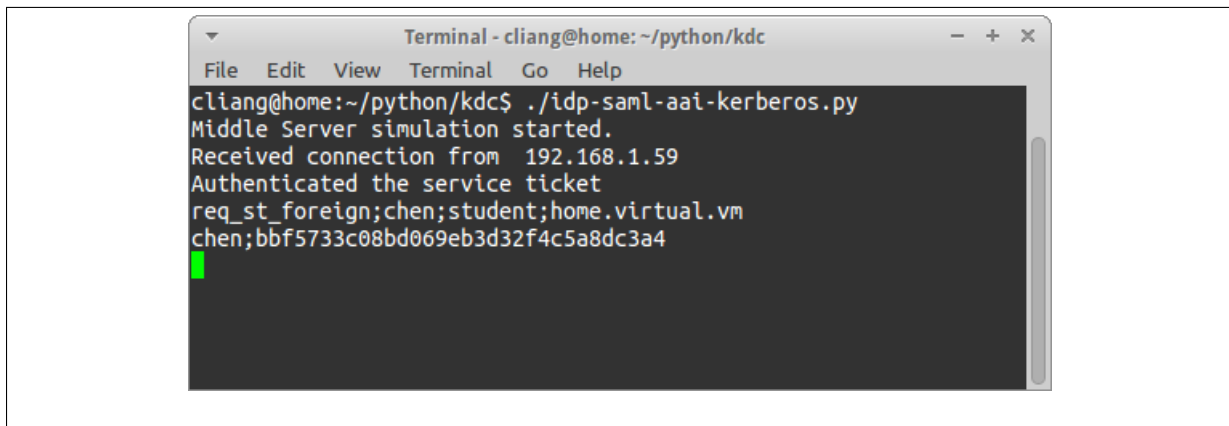


```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp_kdc_sim.py
home
KDC Simulation Server started at: 192.168.1.59
Received connection from 192.168.1.59
Received shared key request.
Shared Key sent.
Received connection from 192.168.1.59
Received TGT request.
Ticket Granting Ticket sent.
Received connection from 192.168.1.59
Received service ticket request.
Service Ticket sent

```

Figure 6.21: KDC simulation in domain home.virtual.vm verifies the end-user ticket granting ticket and issues a service ticket .

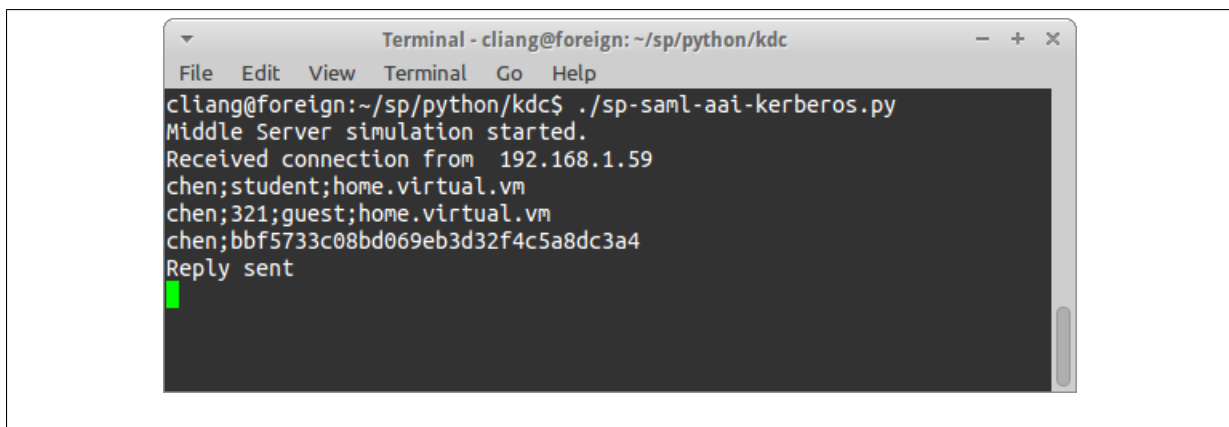


```

Terminal - cliang@home: ~/python/kdc
File Edit View Terminal Go Help
cliang@home:~/python/kdc$ ./idp-saml-aai-kerberos.py
Middle Server simulation started.
Received connection from 192.168.1.59
Authenticated the service ticket
req_st_foreign;chen;student;home.virtual.vm
chen;bbf5733c08bd069eb3d32f4c5a8dc3a4

```

Figure 6.22: Middle-ware system (identity provider) in domain home.virtual.vm verifies the end-user's identity and sends an identity assertion to the middle-ware system (service provide) in domain foreign.virtual.vm .

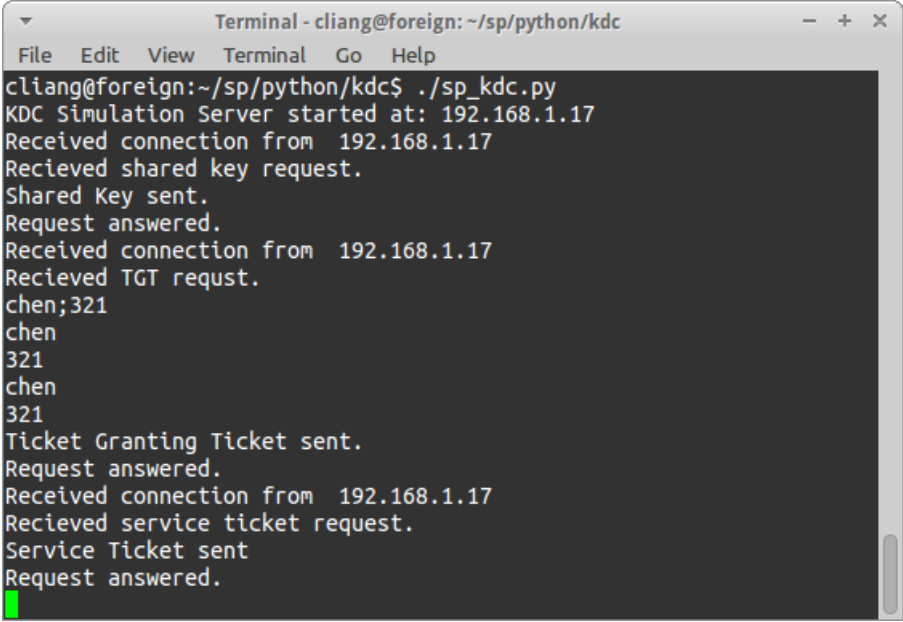


```

Terminal - cliang@foreign: ~/sp/python/kdc
File Edit View Terminal Go Help
cliang@foreign:~/sp/python/kdc$ ./sp-saml-aai-kerberos.py
Middle Server simulation started.
Received connection from 192.168.1.59
chen;student;home.virtual.vm
chen;321;guest;home.virtual.vm
chen;bbf5733c08bd069eb3d32f4c5a8dc3a4
Reply sent

```

Figure 6.23: Middle-ware system (service provider) in domain foreign.virtual.vm accepts the end-user's identity assertion. It then requests ticket granting ticket and service ticket on behalf of the end-user in domain foreign.virtual.vm.

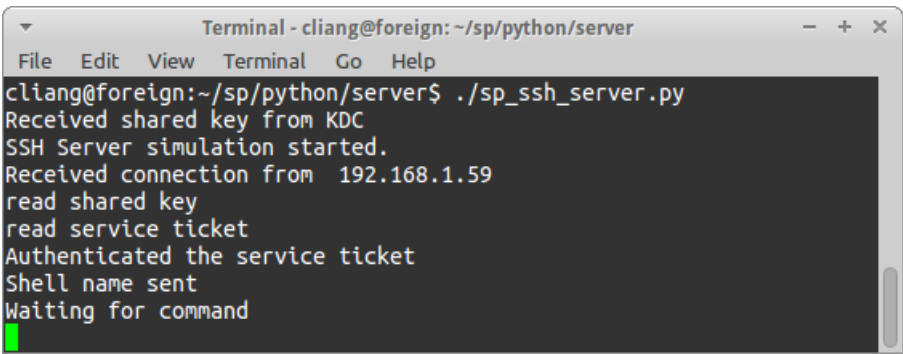


```

Terminal - cliang@foreign: ~/sp/python/kdc
File Edit View Terminal Go Help
cliang@foreign:~/sp/python/kdc$ ./sp_kdc.py
KDC Simulation Server started at: 192.168.1.17
Received connection from 192.168.1.17
Recieved shared key request.
Shared Key sent.
Request answered.
Received connection from 192.168.1.17
Recieved TGT request.
chen;321
chen
321
chen
321
Ticket Granting Ticket sent.
Request answered.
Received connection from 192.168.1.17
Recieved service ticket request.
Service Ticket sent
Request answered.

```

Figure 6.24: KDC simulation in domain foreign.virtual.vm issues ticket granting ticket and service ticket. The service ticket is passed to the end-user in domain home.virtual.vm through the middle-ware system .



```

Terminal - cliang@foreign: ~/sp/python/server
File Edit View Terminal Go Help
cliang@foreign:~/sp/python/server$ ./sp_ssh_server.py
Received shared key from KDC
SSH Server simulation started.
Received connection from 192.168.1.59
read shared key
read service ticket
Authenticated the service ticket
Shell name sent
Waiting for command

```

Figure 6.25: The remote Unix/Linux desktop simulation in domain foreign.virtual.vm accepts the request from the end-user.

```

Terminal - cliang@home: ~/python/client
File Edit View Terminal Go Help
cliang@home:~/python/client$ ./idp_ssh_client_sim.py
Connecting to home ...
Username: chen
Password:
Recieved and saved Ticket Granting Ticket.
local shell~>help
ls      ifconfig      hostname      exit      klist      ssh remot
e.home.virtual.vm      ssh foreign.virtual.vm
local shell~>ssh foreign.virtual.vm
Connecting to home ...
chen;bf6f46c22ce12a9ab2d044a8692cd79e
foreign remote shell~>help
ls      ifconfig      hostname      exit      touch
foreign remote shell~>hostname
foreign
foreign remote shell~>ifconfig
192.168.1.17
foreign remote shell~>ls
./test2
./test3
./test4
./test6
./test5
./test1

foreign remote shell~>exit
exiting...
local shell~>

```

Figure 6.26: The remote Unix/Linux desktop simulation in domain foreign.virtual.vm accepts the request from the end-user. The end-user performed a file creation task.

## 6.4 Summary

This chapter presents the proof of concept implementation of the new simplified federated single sign-on system. Three use cases were demonstrated in user manual, single sign-on in local network, web-based federated single sign-on, and non-web based federated single sign-on. The demonstrations show the new simplified federated single sign-on system successfully allows the end-users to access both web-based and non-web based computer services/applications (in multiple domains) with one authentication session. In addition, it shows the prototype can incorporate network domains' existing authentication infrastructures. As identified in chapter 5, additional functions were required for computer services/applications to distinguish between local network single sign-on and federated single sign-on, therefore, the prototype has not eliminated the need to modify existing computer services/applications.

# Chapter 7

## Research Evaluation

### 7.1 Introduction

This chapter presents the evaluation of the research. The criteria, which are used in the evaluation, were adopted from Project Moonshot and from the author. In the comparison, network domains are assumed to support Kerberos single sign-on (for local network authentication) and SAML based federated single sign-on (for federated authentication).

Section 7.2 presents the criteria development. Section 7.3 uses the developed criteria to evaluate Project Moonshot, SASL-SAML, Kerberos Secure Sharing, SAML-AAI/Kerberos and the approach of this research. Section 7.5 summaries this chapter.

### 7.2 Criteria Development

This chapter presents the evaluation of this research. The evaluation uses comparison criteria of Project Moonshot. While analysing the technical feasibility of Project Moonshot, Painless Security (2010) listed four comparison criteria when looking at proposed solutions to federation of non-web applications. The approach of this research is compared with Project Moonshot, SASL-SAML, Kerberos Secure Sharing, SAML-AAI/Kerberos using these criteria:

- The first criterion is: whether the client needs to talk directly to the authentication infrastructure or whether the conversation with the authentication infrastructure is tunnelled?
- The second criterion for comparison is: how the solution provides protection against phishing and how it provides the mutual authentication service?
- The third criterion is: whether a web browser is used for initial authentication a trusted component of the local environment?
- The fourth criterion for comparison is: how much new development is required and how well the mechanism will fit into existing applications?

In addition to the criteria specified by Painless Security (2010), this research specifies another two criteria:

- The fifth criterion for comparison is: how many authentications are required for an end-user to access two computer services (i.e. web-based and non-web based) in a foreign domain?
- The sixth criterion for comparison is: does the system requires modifications to network domains' existing authentication infrastructures?
- The seventh criterion for comparison is: does the system exposes network domains' existing authentication infrastructures to the outside world?

## 7.3 Evaluation using the Developed Criteria

### 7.3.1 First Criterion

The first criterion is whether the client needs to talk directly to the authentication infrastructure or whether the conversation with the authentication infrastructure is tunnelled (Painless Security, 2010)? Network domains need to change their firewall rules to



compensate direct communications (across multiple domains), therefore, tunnelling communication is more desirable with communicating beyond local network domain. The approaches of this research, Project Moonshot, Kerberos Secure Shearing, SASL-SAML and SAML-AAI/Kerberos are evaluated in table 7.1:

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing uses Kerberos' cross realm authentication, it requires all the Kerberos key distribution centres (KDCs) to be registered with each other (i.e. KDC in each domain has the information of the KDCs in the other domains). In this approach, user agents (i.e. clients) communicate with KDCs directly, and KDCs communicate with each other directly.
<b>Project Moonshot</b>	Project Moonshot uses EAP for federated single sign-on. EAP uses tunnel communication between clients and authentication infrastructures.
<b>SASL-SAML</b>	SASL-SAML uses SASL to modify end-users' user agents, so that they can communicate with SAML based federated single sign-on system (more specifically identity provider) using web communication protocol (e.g. HTTPS). Clients communicate with authentication infrastructure with neither direct communication or tunnel communication.
<b>SAML-AAI/Kerberos</b>	SAML-AAI/Kerberos specified a system that allows web-based services to request Kerberos tickets on behalf of the end-users. It uses SAML based federated single sign-on system (i.e. web communication protocol). Similar to SASL-SAML, it uses neither communication type.
<b>A new Simplified FSSO System</b>	The new simplified federated single sign-on system (this research) uses Kerberos as the authentication system. The user agents (i.e. clients) communicate directly with the KDCs. However, the difference between KSS and this approach is that, this approach does not use Kerberos for federated single sign-on. It uses middle-ware systems to manage federated single sign-on, the middle-ware systems communicate directly with each other.

Table 7.1: Comparing approaches with the first criterion

### 7.3.2 Second Criterion

The second criterion for comparison is how the solution provides protection against phishing and how it provides the mutual authentication service (Painless Security, 2010)? Unauthorised obtaining of end-user's identities can result in catastrophic security issues (e.g. illegal access to end-users' systems and files). This criterion compares the defences employed by listed approach against phishing (Table 7.2).

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing employees symmetrical encryption methods. Kerberos components such as ticket granting ticket and service ticket are encrypted during transaction. According to Painless Security (2010), Kerberos exploits pre-existing relationships to provide their mutual authentication and phishing defence. It may reduce the effectiveness of phishing attacks.
<b>Project Moonshot</b>	Project Moonshot uses EAP, which supports security technologies such as TLS and PKI, for federated authentication. TLS and PKI can defend against phishing attacks at a technical level. However, in practice, these technical defence do not work: end-users are too willing to accept invalid certificates or to ignore other indicators (Painless Security, 2010).
<b>SASL-SAML</b>	As we specified before, the communication between SAML based identity provider and the user agents uses web communication protocols. Similar to Project Moonshot, TLS and PKI can be used to defend against phishing, however, it can not stop end-users from accepting invalid certificates.
<b>SAML-AAI/Kerberos</b>	Similar to SASL-SAML, SAML-AAI/Kerberos uses web-based federated single sign-on. It faces the same issues as Project Moonshot, end-users are too willing to ignore secure for usage.

<b>A new Simplified FSSO System</b>	The new simplified federated single sign-on system (this research) uses Kerberos as the authentication system. This research assumes that the encryption methods are employed, this means the new simplified federated single sign-on system uses the same phishing defence as Kerberos.
-------------------------------------	--

Table 7.2: Comparing approaches with the second criterion

### 7.3.3 Third Criterion

The third criterion is whether a web browser is used for initial authentication or a trusted component of the local environment (Painless Security, 2010)? Many websites have indicated a strong requirement to control the user interface of the login experience. For example in the SAML based federated single sign-on approach, the service provider (SP) presents a web interface. It chooses when or if an identity provider is selected, and it manages the selection of that identity provider. The identity provider (IdP) manages the interface used to authenticate the end-users.

In contrast to a web browser, a trusted component (i.e. user agent) of the local network can manage the federated authentication process. Using a local component provides less opportunity for service parties to manage the interactions. This may cause the end-user experience to be perceived as less cohesive. Support costs may be increased because even though the end-users believe they are logging into the service provider, the service provider had no control over the error messages and may have difficulty documenting all the interactions. This can be especially true if there are multiple different user agents an end-user may have with different user interfaces. However, there are significant advantages to using a local component for authentication. Doing so makes it significantly easier to improve the cryptographic security of the credentials used.

Kerberos secure sharing (KSS) and the new simplified federated single sign-on system consider logging into the desktop (i.e. authentication with network directory in

the local area network) as initial authentication; federated authentication comes after this stage. In contrast, Project Moonshot, SASL-SAML and SAML-AAI/Kerberos consider the first federated authentication as the initial authentication (i.e. they do not counting the desktop authentication as part of the single sign-on process). The third criterion will be used to compare the technology used for the initial federated authentication (Table 7.3).

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing (KSS) uses Kerberos server (i.e. trusted component of the local network) for initial federated authentication.
<b>Project Moonshot</b>	Project Moonshot uses a trusted component of the local network for initial federated authentication.
<b>SASL-SAML</b>	SASL-SAML uses a trusted component of the local network for initial authentication.
<b>SAML-AAI/Kerberos</b>	SAML-AAI/Kerberos uses web browser to access both web-based and non-web services. This approach uses web browser for initial federated authentication.
<b>A new Simplified FSSO System</b>	Similar to KSS, this approach uses Kerberos server (i.e. a trusted component) for initial federated authentication.

Table 7.3: Comparing approaches with the third criterion

### 7.3.4 Fourth Criterion

The fourth criterion for comparison is how much new development is required and how well the mechanism will fit into existing applications (Painless Security, 2010)? The primary goal of this research is a system that is easy to deploy, less modification to existing applications means easier to deploy to network domains. Table 7.4 presents the comparison of Project Moonshot, SASL-SAML, Kerberos Secure Sharing, SAML-AAI/Kerberos and the new simplified federated single sign-on system.

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing uses Kerberos as it's primary authentication system. Since Kerberos has been established as the predominant single sign-on system (Kerberos support has been built in Microsoft Windows systems, GNU/Linux and Mac OS X), it requires little development before deployed into a domain.
<b>Project Moonshot</b>	Project Moonshot proposes to modified the existing user agents to use EAP for federated single sign-on. This means all non-web based applications creators needs to modify their existing user agents.
<b>SASL-SAML</b>	SASL-SAML is similar to Project Moonshot. It modifies the existing user agents to communicate with web-based identity provider through SASL.
<b>SAML-AAI/Kerberos</b>	SAML-AAI/Kerberos specified a system that allows web-based services to request Kerberos tickets on behalf of the end-users. This approach requires the creation of new web-based applications to access non-web based service applications.
<b>A new Simplified FSSO System</b>	This approach needs add additional functions to applications so that they can distinguish between local network single sign-on and cross domain single sign-on (i.e. federated single sign-on).

Table 7.4: Comparing approaches with the fourth criterion

### 7.3.5 Fifth Criterion

This research proposes the fifth criterion for comparison, it is how many authentication stages are required for an end-user to access two computer service (web-based and non-web based) in a foreign domain? As we described before, unlike Project Moonshot, SASL-SAML and SAML-AAI/Kerberos, this research considers that the desktop authentication stage is the initial authentication stage. Table 7.5 compares these approaches in the context of this research.

<b>Kerberos Secure Sharing</b>	Kerberos has been built into the authentication process of Microsoft Windows, GNU/Linux and Mac OS X (10.2.1 and later) operating systems. They have the ability to acquire a Kerberos ticket upon desktop authentication (i.e. login). The end-users can access non-web based services/applications without authentication again. Even though this approach is not aimed to support web-based services/applications, web-based federated single sign-on system can be configured to support Kerberos single sign-on, therefore, one authentication stage is required.
<b>Project Moonshot</b>	Project Moonshot does not consider the desktop authentication as the initial authentication. In this approach, the end-users need to login their desktop first, they need to authenticate again with Project Moonshot's system. In addition, Project Moonshot focus on non-web based service applications. A separate federated single sign-on system is required for web-based service/application. There are in fact three separate authentication stages.
<b>SASL-SAML</b>	Similar to Project Moonshot, SASL-SAML does not consider the desktop authentication as the initial authentication. It requires two authentication stages for federated authentication.
<b>SAML-AAI/Kerberos</b>	Similar to Project Moonshot and SASL-SAML, SAML-AAI/Kerberos requires two authentication stages.
<b>A new Simplified FSSO System</b>	The new simplified federated single sign-on system considers the desktop authenticate as the initial authenticate. Assuming the applications support Kerberos, this approach requires a single authentication stage for an end-user to access multiple computer services in multiple domains.

Table 7.5: Comparing approaches with the fifth criterion

### 7.3.6 Sixth Criterion

The sixth criterion for comparison is: does the system requires modifications to network domains' existing authentication infrastructures? Modifying existing authentication infrastructures can increase the cost of deploying federated single sign-on system, while creating a new authentication infrastructure can create conflicts with the existing ones, therefore, the more authentication infrastructures the approach can incorporate the better. The listed approaches are compared with this criterion in table 7.6.

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing requires network domains to knowl- edge the detail of each others' Kerberos key distribution centres (KDCs). Modifications include registration of KDCs' details (e.g. name and network location) in each other's KDC records and exchanging of secure keys.
<b>Project Moonshot</b>	Project Moonshot does not modify network domains existing authentication infrastructures. It deploys a separate authenti- cation infrastructure for federated single sign-on.
<b>SASL-SAML</b>	SASL-SAML modifies SAML based federated single sign-on system to support non-web based applications.
<b>SAML-AAI/Kerberos</b>	SAML-AAI/Kerberos uses the existing SAML based federated single sign-on system (identity provider). It does not modify network domains' authentication infrastructure on the identity provider side. It modifies the authentication infrastructure on the service provider side, so that service provider can request Kerberos ticket on behalf of end-users.
<b>A new Simplified FSSO System</b>	This approach incorporates network domains' existing authen- tication infrastructures. It does not modify them.

Table 7.6: Comparing approaches with the sixth criterion.

### 7.3.7 Seven Criterion

The seventh criterion for comparison is: does the system exposes network domains' existing authentication infrastructures to the outside world? Network domains are reluctant to use any system that will expose their internal authentication infrastructure to the outside world. The listed approaches are compared with this criterion in table 7.7.

<b>Kerberos Secure Sharing</b>	Kerberos secure sharing requires network domains to share information on their Kerberos key distribution centres. It exposes Kerberos key distribution centres to the outside world.
<b>Project Moonshot</b>	Project Moonshot uses a separate federated single sign-on system. It does not use the existing authentication infrastructure.
<b>SASL-SAML</b>	Similar to Project Moonshot, it does not use the existing authentication infrastructure.
<b>SAML-AAI/Kerberos</b>	SAML-AAI/Kerberos requires service providers to request Kerberos ticket on behalf of the end-users. It does not exposes the Kerberos key distribution centres to the outside world.
<b>A new Simplified FSSO System</b>	The middle-wares facilitate federated singles sign-on, it does not expose existing authentication infrastructure to the outside world.

Table 7.7: Comparing approaches with the seventh criterion.

## 7.4 Comparison Overview

This section presents the side by side comparison of the characteristic of the new simplified federated single sign-on system, Kerberos secure sharing (KSS), Project Moonshot, SASL-SAML and SAML-AAI/Kerberos.

---



	<b>New Simplified FSSO System</b>	<b>Keberos Secure Sharing</b>	<b>Project Moonshot</b>	<b>SASL-SAML</b>	<b>SAML-AAI/Kerberos</b>
Modifications to Firewall	Yes	Yes	No	No	No
Protection Against Fishing	Strong	Strong	Difficult to defend end-users against invalid certificates	Difficult to defend end-users against invalid certificates	Difficult to defend end-users against invalid certificates
Authentication with web browser or applications' native clients	local trusted component	local trusted component	local trusted component	local trusted component	web browser
Modifications to existing computer services/applications	Add addition functions to distinguish local SSO and Federated SSO	None	Add support for new authentication infrastructure	Add support for new authentication infrastructure	Create new web-based application clients
Number of authentication for accessing two (web-based and non-web based) applications	One	One	Three	Two	Two

Modifications to existing authentication infrastructure	None	Extend Kerberos servers beyond network domains	Separate authentication infrastructure	Modify SAML based FSSO system to support non-web based applications	Modify service provider's authentication infrastructure
Exposing existing authentication infrastructure to the outside world	No	Yes	No	No	No

Table 7.8: Table comparison

## 7.5 Summary

This chapter used six comparison criteria to compare the new simplified federated single sign-on system against existing approaches (e.g. Project Moonshot, SASL-SAML, Kerberos secure sharing and SAML-AAI/Kerberos). The evaluation shows that the new simplified federated single sign-on system incorporates the network domains' existing authentication infrastructures, therefore, it is simpler to deploy than the other approaches. Unlike Kerberos secure sharing, it achieved it without exposing network domains' authentication infrastructures to the outside world. In addition, end-users only need to authenticate once to access diverse computer services/applications (in multiple domains). Although, addition functions were added to existing services/applications, so that they can distinguish between local network single sign-on and federated single sign-on, and modifications to firewalls are needed in order to support communications between middlewares. This new simplified federated single sign-on system had made huge process towards developing a new simple to deploy federated single sign-on system.

# Chapter 8

## Discussion and Conclusion

### 8.1 Introduction

The overall objective of this research is to develop a prototype of a new simplified federated single sign-on system. This research found that currently it is not possible for an end-user to authenticate (using federated single sign-on method) to diverse computer services/applications (in multiple domains) with a single authentication session.

Four existing approaches that aim to address this issue: Project Moonshot (Howlett, 2010), SASL-SAML (Wierenga & Lear, 2010) (i.e. the combination Simple Authentication and Security Layer with security assertion markup language), Kerberos Secure Sharing (Gridwise Tech, 2010) and SAML-AAI/Kerberos (Papez, 2009) (i.e. the combination of Security Assertion Markup Language's authentication authorisation infrastructure with Kerberos single sign-on infrastructure). Analysis of these approaches (chapter 4) shows that they requires active buy-in from network domains (e.g. modification to network domains' authentication infrastructure and modification to existing computer services/applications), otherwise these models will fail. There is a lack of simple to deploy federated single sign-on system.

The objective has been satisfied as follow. The new simplified federated single sign-on system allows end-users to access diverse computer services/applications (in mul-

multiple domains) with one authentication session. In addition, it can incorporate network domains' authentication infrastructure. Unlike existing approaches, network domains do not need to modify their existing authentication infrastructures or deploy separate authentication infrastructures to support federated single sign-on. The existing authentication infrastructures are used as part of the new simplified federated single sign-on system.

Although this research had made huge progress on developing a simple to deploy federated single sign-on system. Evaluation shows that additional modifications to the existing computer services/applications were needed, so that they can distinguish between local network single sign-on and federated single sign-on. Similar to existing approaches, network domains need to actively buy-in this model. In addition, modification to firewall were also needed to support middle-ware communications.

This approach relies on middle-ware in network domains. Each domain has one middle-ware for federated single sign-on. It may suffer from single point of failure. This research suggests the use of multiple middle-wares. If one middle-ware that facilitates federated single sign-on fails, another one immediately takes over. This ensures the continuous availability of the system.

Section 8.2 presents the contributions of this research. Section 8.3 presents the future research on the topic. Section 8.4 presents the potential applications to other fields.

## 8.2 Contributions

### 8.2.1 Primary Contribution

The primary contribution of this research is the development of a new simplified federated single sign-on system. It made huge progress towards developing a simple to deploy federated single sign-on system. This approach can incorporate network domains' authentication infrastructures so that network domains do not need to modify their existing authentication infrastructure or deploying a new authentication infrastructure. In ad-

dition, it delivers one authentication session for an end-user to access web-based and non-web based services/applications. For example, once an end-user had been authenticated with Kerberos authentication system, the new simplified federated single sign-on system allows the end-users to access both web-based and non-web based computer services/applications in multiple domains. A prototype of the system has been developed and presented in Chapter 5. The prototype has been implemented and tested in a simulation of real world environment (Chapter 6).

### 8.2.2 Secondary Contributions

The secondary contributions are:

1. The research contributes to the body of knowledge by reviewing the existing approaches and developing the understanding of federated single sign-on systems.
2. A simulation of real world domain environment (i.e. testing environment) is designed and implemented to evaluate the new simplified federated single sign-on system. The simulation follows the research of real world authentication systems (SAML, Kerberos, Unix/Linux desktop, Secure Shell and web-based service). It is contributed as evaluation system to the research of federated single sign-on.

## 8.3 Future Work

As presented in the evaluation, the new simplified federated single sign-on system still presents the following issues: existing computer services/applications required modifications so that they can distinguish between local network single sign-on and federated single sign-on, firewalls also required modifications so that they can support middle-ware communications, and encryption and decryption of Kerberos tickets may require both network domains to exchange their secure key (through the middle-ware). The detailed future work are presented as the following:

1. Further research on deploying federated single sign-on system without modifying existing computer services/applications.
2. Investigate the compatibility between the Kerberos encryption technologies and the approach of this research.
3. Research on minimising modifications to firewalls.

### **Future work 1**

Currently system still requires modifications to existing computer services/applications and it remains a obstacle to system deployment. It is important to continue the research and find a way to eliminate the modifications, hence, network domains do not need to active buy-in this approach.

### **Future work 2**

Different network domains may use different encryption keys for encrypting and decrypting Kerberos tickets. It is important to continue the research on Kerberos' encryption technologies in the new simplified federated single sign-on system (e.g. middle-ware system can be used to transfer secure keys between domains). This will ensure that the Kerberos tickets can be successfully transfer between domains, and they can be used by end-users to access computer services in other domains.

### **Future work 3**

The evaluation shows that modifications are required for firewalls to support the communications between middle-wares. To future simplified the deployment of this approach, it is important to investigate methods (e.g. tunnelling network) that minimise the modifications to the firewalls.

## 8.4 Potential Application of the new Simplified FSSO System

### 8.4.1 Federated Single Sign-On in Cloud Computing

Cloud computing is a service oriented computing mode that enables convenient and on-demand network access to highly scalable and configurable computing resources (NIST, 2010; Mell & Grance, 2011). It contains multiple cloud service models, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). SaaS provides computer applications that are hosted and managed by Cloud vendors, PaaS provides application development environments hosted and managed by Cloud vendors, and IaaS provides computer infrastructures hosted by the Cloud vendors. These service models provide diverse cloud services ranging from web-based applications to desktop sessions, which is similar to web-based and non-web environment in this research. In addition, Cloud vendors use their own separate domain. Multiple Cloud vendors means multiple domains, it also means Cloud vendors can come together to form a federation. By extending the new simplified federated single sign-on system into Cloud space, an end-user of one Cloud vendor can access computer services in the other Cloud with a single authentication session.

# References

- Local and Metropolitan Area Networks: Port-Based Network Access Control, IEEE Standard 802.1X.
- Standard for Local and Metropolitan Area Networks: Overview and Architecture, IEEE Standard 802, 1990.
- Unapproved Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security, IEEE Draft 802.11i (work in progress).
- Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and Levkowetz, E. H., 2004. Extensible Authentication Protocol (EAP). *Request for Comments: 3748*, .
- Ahn, G.-J. and Ko, M., 2007. User-centric Privacy Management for Federated Identity Management. In *Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on*, pages 187 –195.
- Anderson, J., 1973. Information Security in a Multi-User Computer Environment. *Advances in Computers*, **12**:1–35.
- Balasubramaniam, S., Lewis, G., Morris, E., Simanta, S., and Smith, D., 2009. Identity Management and Its Impact on Federation in a System-of-systems Context. In *Systems Conference, 2009 3rd Annual IEEE*, pages 179 –182.
- Benantar, M., 2005. *Access Control Systems: Security, Identity Management and Trust Models*. Springer.



- Bertino, E. and Takahashi, K., 2010. *Identity Management: Concepts, Technologies, and Systems (Information Security & Privacy)*. Artech House Publishers.
- Bhargav-Spantzel, A., Camenisch, J., Gross, T., and Sommer, D., 2007. User centrality: A Taxonomy and Open Issues. *Journal of Computer Security*, **15**.
- Bishop, M., 2003. What is computer security? *Security Privacy, IEEE*, **1**(1):67 – 69.
- Bishop, M., 2005. *Introduction to Computer Security*. Addison-Wesley.
- Blezard, D. J. and Marceau, J., 2002. One user, One Password: Integrating Unix Accounts and Active Directory. :5–8.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., and E. Maler, F. Y., 2008. *Extensible Markup Language (XML) 1.0*. W3C Recommendation, <http://www.w3.org/TR/xml/>., fifth edition edition.
- Carter, G., 2003. *LDAP System Administration*. O'Reilly.
- Cerf, V., Dalal, Y., and Sunshine, C., 1974. Request for Comments (Network Working Group) (Electronic text) No. 675. 1 (title) Specification of internet transmission control program. .
- Chappell, D., 2006. Introducing Windows CardSpace. *Retrieved from <http://msdn.microsoft.com/en-us/library/Aa480189> in 2012*, .
- Chokhani, S. and Ford, W., 1999. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. *Request for Comments: 2527, Retrieved from <http://www.ietf.org/rfc/rfc2527.txt> in 2012*, .
- Corbato, F., Daggett, M., Daley, R., Creasy, R., Hellwig, J., Orestein, R., and Korn, L., 1963. The Compatible Time-Sharing System A Programmer's Guide. .
- Corbato, F., Merwin-Daggett, M., and Daley, R., 1992. CTSS - The Compatible Time-sharing System. *Annals of the History of Computing, IEEE*, **14**(1):31 –32.

- Denning, D. E. and Sacco, G. M., 1981. Timestamps in Key Distribution Protocols. *Commun. ACM*, **24**:533–536.
- Don and Smith, 2008. The Challenge of Federated Identity Management. *Network Security*, **2008**(4):7 – 9.
- Dunleavy, P., Margetts, H., Bastow, S., and Tinkler, J., 2006. Digital Era Governance. *Oxford University Press*, :251.
- El-Hadidi, M., Hegazi, N., and Aslan, H., 1997. Performance analysis of the Kerberos protocol in a distributed environment. In *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pages 235 –239.
- El Maliki, T. and Seigneur, J.-M., 2007. A Survey of User-centric Identity Management Technologies. In *Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007. The International Conference on*, pages 12 –17.
- Ellison, C. and Schneier, B., 2000. Ten Risks of PKI: What You’re not Being Told about Public Key Infrastructure. *COMPUTER SECURITY JOURNAL*, **16**:1–8.
- Glasser, U. and Vajihollahi, M., 2008. Identity Management Architecture. In *Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on*, pages 137 –144.
- Godik, S. and Moses, T., 2003. eXtensible Access Control Markup Language (XACML) Version 1.1. Technical report, Organization for the Advancement of Structured Information Standards.
- Google Inc., 2012. SAML Single Sign-On (SSO) Service for Google Apps - [https://developers.google.com/google-apps/sso/saml\\_reference\\_implementation](https://developers.google.com/google-apps/sso/saml_reference_implementation). Technical report.
- Gridwise Tech, 2010. Kerberos: Secure Sharing of Distributed Resources. Technical report.
- Haller, N. and Metz, C., 1996. A One-Time Password System. *RFC 1938*, .

- Halperin, R. and Backhouse, J., 2008. A Roadmap for Research on Identity in the Information Society. *Identity in the Information Society*, **1**:71–87. 10.1007/s12394-008-0004-0.
- HEAnet, 2008. Board Approves New 5-year Strategic Plan. Technical report.
- HEAnet, 2012a. Edugate Members - Retrived from <http://www.edugate.ie/content/edugate-federation-members>. Technical report, Edugate.
- HEAnet, 2012b. HEAnet, Ireland's National Education & Research Network. *Retrieved from <http://www.heanet.ie> in 2012*, .
- Hedin, F. H., 1937. Electrical Computer Eliminates Calculations. *American Institute of Electrical Engineers, Transactions of the*, **56**(7):787 –790.
- Hinsley, F. and Stripp, A., 2001. *Codebreakers: the inside story of Bletchley Park*. Oxford University Press.
- Howes, T. and Smith, M., 1997. *LDAP: Programming Directory-enabled Applications with Lightweight Directory Access Protocol*. Sams Publishing.
- Howes, T., Smith, M., and Good, G. S., 2003. *Understanding and Deploying LDAP Directory Services*. Addison-Wesley, 2nd edition.
- Howlett, J., 2010. Project Moonshot. *Briefing paper for IEFT, Maastricht 2010*, .
- Howlett, J., 2011. Project Moonshot. *Retrieved from <http://www.project-moonshot.org/>*, .
- Hunter, L. E., 2006. *Active Directory Cookbook*. O'Reilly, 2nd edition.
- IANA, 2012. Internet Assigned Numbers Authority (IANA). *Retrieved from <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml>*, .
- Internet2, 2012. Shibboleth. *Retrieved from <http://shibboleth.internet2.edu/>*, .

- Jain, A., Bolle, R., and Pankanti, S., 1999. *Biometrics: personal identification in networked society*. The Kluwer international series in engineering and computer science. Kluwer.
- JANET, 2012. JANET, the UK's education and research network. *Retrieved from <http://www.ja.net/> in 2012, .*
- Jason and Goode, 2012. The Importance of Identity Security. *Computer Fraud & Security*, **2012**(1):5 – 7.
- Jøsang, A. and Pope, S., 2005. User Centric Identity Management. *AusCERT Conference*, .
- Jøsang, A., Zomai, M. A., and Suriadi, S., 2007. Usability and Privacy in Identity Management Architectures. In *Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68*, ACSW '07, pages 143–152, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.
- Kilburn, T., Howarth, D. J., Payne, R. B., and Sumner, F. H., 1961. The Manchester University Atlas Operating System. *The Computer Journal*, **4**(3):222–225.
- Kohl, J. T., Neuman, B. C., and Ts'o, T. Y., 1991. The Evolution of the Kerberos Authentication Service. pages 78–94. IEEE Computer Society Press.
- Lee, H., Jeun, I., and Jung, H., 2009. Criteria for Evaluating the Privacy Protection Level of Identity Management Services. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on*, pages 155 –160.
- Licklider, Robert, Licklider, J. C. R., and Taylor, R. W., 1968. The Computer as a Communication Device. *Science and Technology*, **76**:21–31.
- Linn, J., 2000. Generic Security Service Application Program Interface Version 2, Update 1. *Retrieved from <http://tools.ietf.org/html/rfc2743> in 2012, .*

- MacKenzie, D. and Pottinger, G., 1997. Mathematics, technology, and trust: formal verification, computer security, and the U.S. military. *Annals of the History of Computing, IEEE*, **19**(3):41 –59.
- Madsen, P., Maler, E., Wisniewski, T., Nadalin, T., Cantor, S., Hodges, J., and Mishra, P., 2005. SAML V2.0 Executive Overview. .
- Maler, E., Mishra, P., and Philpot, R., 2003. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. *OASIS Standard*, .
- Maler, E. and Reed, D., 2008. The Venn of Identity: Options and Issues in Federated Identity Management. *Security Privacy, IEEE*, **6**(2):16 –23.
- Mather, T., Kumaraswamy, S., and Latif, S., 2009. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Series. O'Reilly Media.
- Matsumoto, T., Matsumoto, H., Yamada, K., and Hoshino, S., 2002. Impact of Artificial Gummy Fingers on Fingerprint Systems. *SPIE*, **4677**.
- McRae, M., 2009. Approval of WS-Federation v1.2 as an OASIS Standard. Technical report, OASIS.
- Mell, P. and Grance, T., 2011. The NIST Definition of Cloud Computing. *National Institute of Standard and Technology*, :2.
- Melnikov, A. and Zeilenga, K., 2006. Simple Authentication and Security Layer. *Request for Comments: 4422*, .
- Microsoft, 2012a. Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol Specification. Technical report, Retrieved at <http://msdn.microsoft.com/en-us/library/cc246071>
- Microsoft, 2012b. WS-Federated Authentication Module Overview. Retrieved from <http://msdn.microsoft.com/en-us/library/ee517293.aspx>, .

- Migeon, J.-Y., 2008. The MIT Kerberos Administrators How-to Guide. *MIT Kerberos Consortium*, :62.
- Miller, S. P., Neuman, B. C., Schiller, J. I., and Saltzer, J. H., 1988. Kerberos Authentication and Authorization System. In *In Project Athena Technical Plan*.
- MIT, 2008. Best Practices for Integrating Kerberos into Your Application. *MIT Kerberos Consortium*, .
- Miyata, T., Koga, Y., Madsen, P., ichi Adachi, S., Tsuchiya, Y., Sakamoto, Y., and Takahashi, K., 2006. A Survey on Identity Management Protocols and Standards. *IEICE TRANSACTIONS on Information and Systems*, **E89-D No.1**:112–123.
- Mont, M. C., Bramhall, P., Gittler, M., Pato, J., and Rees, O., 2002. Identity Management: a Key e-Business Enabler. .
- Mont, M. C., Bramhall, P., and Pato, J., 2003. On Adaptive Identity Management: The Next Generation of Identity Management Technologies. .
- Needham, R. M. and Schroeder, M. D., 1978. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, **21**:993–999.
- Newham, C. and Rosenblatt, B., 2005. *Learning the bash shell, third edition*. O'Reilly Media, Inc., 3 edition.
- NIST, 2010. National Institute of Standards and Technology. *Retrieved from <http://www.nist.gov/>*, .
- OASIS, 2010. SAML OASIS Standard. *Retrieved from <http://saml.xml.org>*, .
- OASIS, 2011. Organization for the Advancement of Structured Information Standards. *Retrieved from <http://www.oasis-open.org/home/index.php>*, .
- Oxford English Dictionary, 2004. *Operating System, n.; Third edition, June 2004; on-line version December 2011. <http://0-www.oed.com.ditlib.dit.ie/view/Entry/131747>; accessed 24 January 2012. An entry for this word was first included in New English Dictionary, 1903*. Oxford University Press.

- Oxford English Dictionary, 2008. *Computer n.*; *Oxford English Dictionary*; Third edition, March 2008; online version December 2011. <http://0-www.oed.com.ditlib.dit.ie/view/Entry/37975>; accessed 24 January 2012. An entry for this word was first included in *New English Dictionary*, 1891. Oxford University Press, online version edition.
- Oxford English Dictionary, 2010. *role, n.*; *Oxford English Dictionaries*; Third edition, September 2010; online version March 2012. <http://0-www.oed.com.ditlib.dit.ie/view/Entry/166971>; accessed 16 May 2012. An entry for this word was first included in *New English Dictionary*, 1909. Oxford University Press.
- Oxford English Dictionary, 2011. *software, n.*; *Oxford English Dictionaries*; Second edition, 1989; online version December 2011. <http://0-www.oed.com.ditlib.dit.ie/view/Entry/183938>; accessed 26 January 2012. First published in *A Supplement to the OED IV*, 1986. Oxford University Press.
- Painless Security, 2010. Project Moonshot: Feasibility Analysis. .
- Papez, R., 2009. SAML-AAI/Kerberos integration.
- Patterson, P., Thiagarajan, P. V., and Sum, M., 2004. Federated Identity: Single Sign-On Among Enterprises. :1–12.
- Paul and Madsen, 2004. Federated Identity and Web Services. *Information Security Technical Report*, **9**(3):56 – 65.
- Pfleeger, C. P. and Pfleeger, S. L., 2006. *Security in Computing*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 4th edition.
- Ping Identity, 2012. Pingfederate - retrieved from <https://www.pingidentity.com/products/pingfederate/>. Technical report.
- Project, L. A., 2003. Introduction to the Liberty Alliance Identity Architecture. .

- Ragouzis, N., Hughes, J., Philpott, R., Maler, E., Madsen, P., and Scavo, T., 2008. Security Assertion Markup Language (SAML) V2.0 Technical Overview. *Committee Draft 02*, .
- Rashid, F. Y., Accessed on Jan 18th 2012. Zappos Breach Illustrate the Need for Stronger Password Rules. <http://www.eweek.com/c/a/Security/Zapps-Breach-Illustrate-the-Need-for-Stronger-Password-Rules-672979/?kc=rss>, .
- Richards, J., Allen, R., and Lowe-Norris, A. G., 2006. *Active Directory*. O'Reilly, 3rd edition.
- Salomon, D., 2005. *Foundations of Computer Security*. Springer; 1st Edition.
- Saltzer, J. and Schroeder, M., 1975. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, **63**(9):1278 – 1308.
- Samba Team, 2006. *Practical Exercises in Successful Samba Deployment*.
- Sato, H. and Nishimura, T., 2011. Federated Authentication in a Hierarchy of IdPs by Using Shibboleth. In *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, pages 327 –332.
- Schuman, E., 2006. Consumers Resist Retail Biometrics. *eWeek*, .
- ScienceDirect, 2003. Gov. Agencies join Liberty Alliance. *Computer Fraud & Security*, **2003**(4):3 –.
- Seigneur, J.-M., 2005. Trust, security and privacy in global computing Ph.D. thesis. *Dublin: Trinity College*, .
- Shim, S., Bhalla, G., and Pendyala, V., 2005. Federated Identity Management. *Computer*, **38**(12):120 – 122.
- Simpson, W., 1994. The Point-to-Point Protocol (PPP). *Request for Comments: 1661*, .
- Smith, R. E., 2001. *Authentication: From Passwords to Public Keys*. Addison-Wesley Professional, 1st edition edition.



- Solomon, D., 1998. The Windows NT Kernel Architecture. *Computer*, **31**(10):40–47.
- Stallings, W., 2006. *Cryptography and Network Security: Principles and Practice*. The William Stallings Books on Computer and Data Communications. Pearson/Prentice Hall.
- Sullivan, R. K., 2005. The Case for Federated Identity. *Network Security*, **2005**(9):15 – 19.
- Sun Microsystems, 2004. Securing your Business Performance Management (BPM) Environment with Identity Solutions from Sun Microsystems. Retrieved from <http://developers.sun.com/identity/reference/whitepapers/hyperion.pdf> in 2012, .
- Suriadi, S., Foo, E., and Jøsang, A., 2009. A User-centric Federated Single Sign-on System. *Journal of Network and Computer Applications*, **32**(2):388 – 401.
- SWITCHaai, 2012. *Shibboleth Demo*. Swiss Education & Research Network.
- Takaaki, K., Hiroaki, S., Noritoshi, D., and Ken, M., 2011. Design and Implementation of Web Forward Proxy with Shibboleth Authentication. In *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, pages 321 –326.
- Thomas, I. and Meinel, C., 2009. Enhancing Claim-Based Identity Management by Adding a Credibility Level to the Notion of Claims. In *Services Computing, 2009. SCC '09. IEEE International Conference on*, pages 243 –250.
- Vacca, J. R., Seigneur, J.-M., and Maliki, T. E., 2010. *Managing Information Security*. Syngress Media.
- Venkataramappa, V., 2002. Single sign on for kerberos authentication.
- Voglmaier, R. E., 2003. *The ABCs of LDAP: How to Install, Run, and Administer LDAP Services*. CRC Press.
- Walden, D. and Vleck, T. V., 2011. The Compatible Time Sharing System (1961-1973) Fiftieth Anniversary Commemorative Overview. *The IEEE Computer Society History*, .

- Wang, X., Schulzrinne, H., Kandlur, D., and Verma, D., 2008. Measurement and Analysis of LDAP Performance. *IEEE/ACM Trans. Netw.*, **16**(1):232–243.
- Wason, T., Cantor, S., Hodges, J., Kemp, J., and Thompson, P., 2003. Liberty ID-FF Architecture Overview. Version: 1.2-errata-v1.0. Retrieved from <http://projectliberty.org/liberty/content/download/318/2366/file/draft-liberty-idff-arch-overview-1.2-errata-v1.0.pdf> in 2012, .
- Wierenga, K. and Lear, E., 2010. A SASL Mechanism for SAML. *Internet Engineering Task Force (IETF)*, .
- William and Knight, 2004. IBM Joins Liberty Alliance. *Infosecurity Today*, **1**(6):10 –.
- Windley, P., 2005. *Digital Identity*. O'Reilly Series. O'Reilly.
- Ying, W., 2010. Research on Multi-level Security of Shibboleth Authentication Mechanism. In *Information Processing (ISIP), 2010 Third International Symposium on*, pages 450 –453.
- Ylonen, T. and Lonvick, C., 2006. The Secure Shell (SSH) Authentication Protocol. *Internet Official Protocol Standards*, .
- Zhang, J., Lang, B., and Duan, Y., 2011. An XML Data Placement Strategy for Distributed XML Storage and Parallel Query. In *Parallel and Distributed Computing, Applications and Technologies (PDCAT), 2011 12th International Conference on*, pages 433 –439.

# Appendix A

## System Specifications

### A.1 Hardware Specifications

The new simplified federated single sign-on system was developed on Sun Ultra 40M2 Workstation and HP-ProBook 6550b (Table A.1). The hardware specifications are as following:

In addition to the hardware that were used for developing the system. The virtual machines' control centre, XenCenter 6.0, was installed on a Dell Optiplex GX260. Hardware includes 2.40 GHz Intel(R) Pentium(R) 4 CPU, 512 MB RAM, Intel(R) PRO/1000MT Network Connection and Intel(R) 82845G Graphic Controller. A Linksys Wireless-N Gigabit Router (WRT350N) is used to create network connections between applications and machines. The router issues IP addresses to all of the machines (physical and virtual). The router has the IP address 192.168.1.1. The IP address range for all the machines is from 192.168.1.10 to 192.168.1.254.

### A.2 Software Specifications

	<b>Sun Ultra 40M2 Workstation</b>	<b>HP-ProBook 6550b</b>
Processor	2 x Dual-Core AMD Opteron (tm) Processor 2222	4 x Intel(R) Core(TM) i3 CPU M 370
Memory	4096MB	2048MB
Hard disk drive (HDD)	128GB	300GB
Networking	2 x Bridge: nVidia Corporation MCP55 Ethernet	<ul style="list-style-type: none"> <li>• Intel Corporation 82577LC Gigabit Network Connection</li> <li>• Broadcom Corporation BCM4313 802.11b/g/n Wireless LAN Controller</li> </ul>
Video	nVidia Corporation G92 [Quadro FX 3700]	Intel Corporation Core Processor Integrated Graphics Controller

Table A.1: IBM eServer Hardware Overview.

	<b>DNS server</b>	<b>Beta Virtual Server</b>
Processor	Single CPU	Single CPU
Memory	512MB	1024MB
Hard disk drive (HDD)	8GB	8GB
Networking	Bridge Connection	Bridge Connection
Operating System	Ubuntu Server 10.04.4 LTS 32bit	Ubuntu Server 10.04.4 LTS 32bit
Key Software	<ul style="list-style-type: none"> <li>• bind9</li> <li>• openssh-server</li> </ul>	<ul style="list-style-type: none"> <li>• apache2</li> <li>• php5</li> <li>• openssh-server</li> <li>• python2.6</li> </ul>
IP Address	192.168.1.24	192.168.1.17
Domain Name	hostmaster.virtual.vm	foreign.virtual.vm
MAC	CA:78:6B:9B:9B:5D	B2:CD:11:E2:F5:87

Table A.2: Sun Ultra 40M2 Workstation Configuration.

Operating System	<b>DNS server</b> Xubuntu Desktop 12.04 LTS 64bit
Key Software	<ul style="list-style-type: none"> <li>• apache2</li> <li>• php5</li> <li>• python2.7</li> </ul>
IP Address	192.168.1.59
Domain Name	home.virtual.vm
MAC	AC:81:12:43:58:7C

Table A.3: HP Probook 6550b Software Configuration.

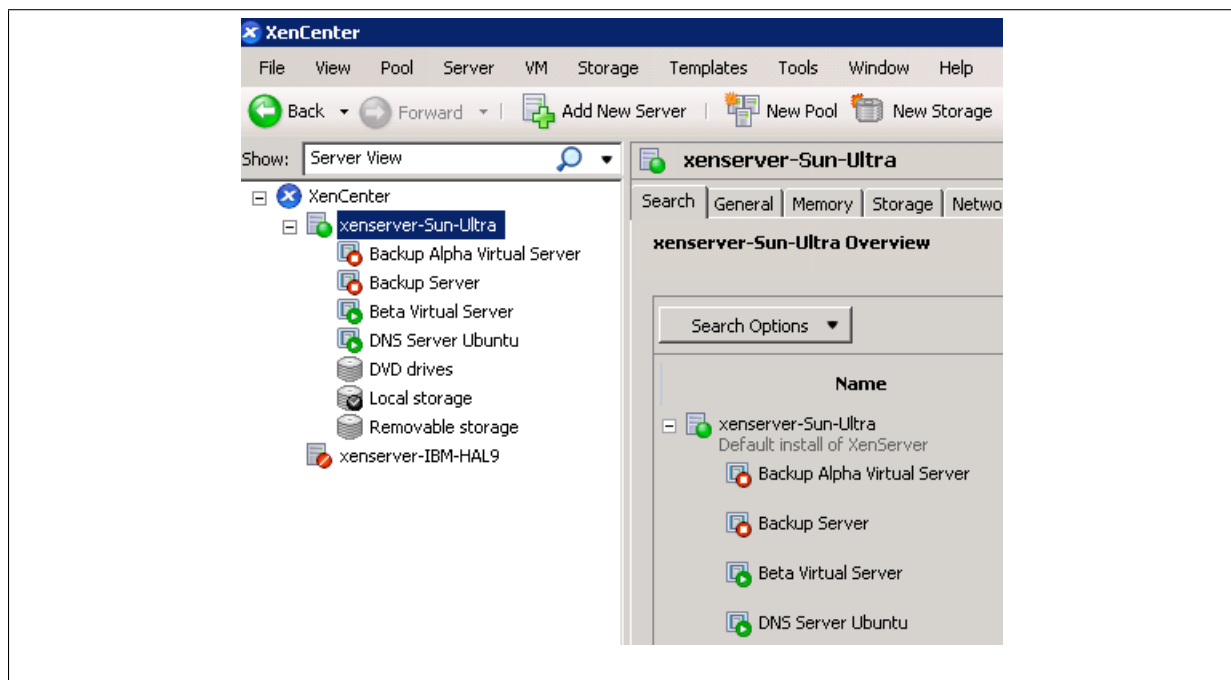


Figure A.1: Dell Optiplex GX260 configuration.

# Appendix B

## List of Security Technologies

### B.1 The Simple Authentication and Security Layer (SASL)

The Simple Authentication and Security Layer (SASL)(Melnikov & Zeilenga, 2006) is a framework for providing authentication and data security services in connection-oriented protocols via replaceable mechanisms. SASL provides a structured interface between protocols and mechanisms. SASL also provides a protocol for securing subsequent protocol exchanges within a data security layer. The data security layer can provide data integrity, data confidentiality, and other services.

SASL's design is intended to allow new protocols to reuse existing mechanisms without requiring redesign of the mechanisms and allows existing protocols to make use of new mechanisms without redesign of protocols.

SASL is conceptually a framework that provides an abstraction layer between protocols and mechanisms as illustrated in figure B.1.

It is through the interfaces of this abstraction layer that the framework allows any protocol to utilize any mechanism. While this layer does generally hide the particulars of protocols from mechanisms and the particulars of mechanisms from protocols, this layer does not generally hide the particulars of mechanisms from protocol implementations.

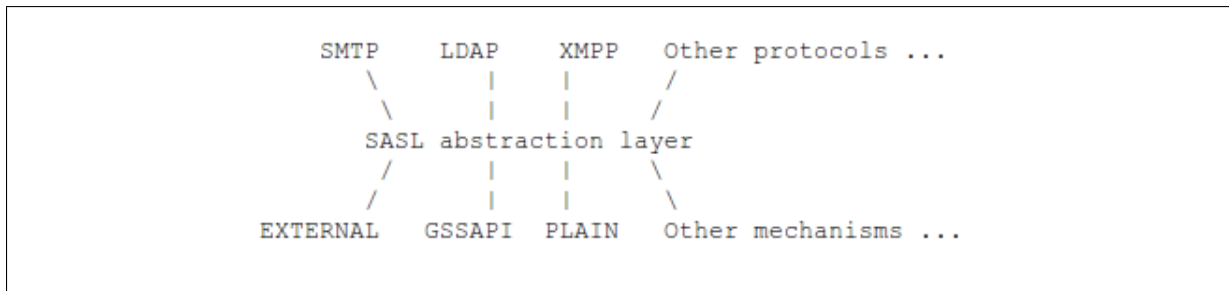


Figure B.1: SASL is conceptually a framework that provides an abstraction layer between protocols and mechanisms(Melnikov & Zeilenga, 2006)

For example, different mechanisms require different information to operate, some of them use password-based authentication, some of them require realm information, others make use of Kerberos tickets, certificates, etc. Also, in order to perform authorization, server implementations generally have to implement identity mapping between authentication identities, whose form is mechanism specific, and authorization identities, whose form is application protocol specific.

## B.2 Generic Security Service Application Program Interface (GSS-API)

Generic Security Service Application Program Interface (GSS-API)(Linn, 2000) Version 2 provides security services to callers in a generic fashion, supportable with a range of underlying mechanisms and technologies and hence allowing source-level portability of applications to different environments. This specification defines GSS-API services and primitives at a level independent of underlying mechanism and programming language environment, and is to be complemented by other, related specifications:

- documents defining specific parameter bindings for particular language environments.
- documents defining token formats, protocols, and procedures to be implemented in order to realize GSS-API services atop particular security mechanisms.

### B.3 Extensible Authentication Protocol (EAP)

Extensible Authentication Protocol (EAP)(Aboba et al., 2004) is an authentication framework which supports multiple authentication methods. EAP typically runs directly over data link layers such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP. EAP provides its own support for duplicate elimination and retransmission, but is reliant on lower layer ordering guarantees. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this.

EAP may be used on dedicated links, as well as switched circuits, and wired as well as wireless links. To date, EAP has been implemented with hosts and routers that connect via switched circuits or dial-up lines using PPP(Simpson, 1994). It has also been implemented with switches and access points using IEEE 802(IEE, b). EAP encapsulation on IEEE 802 wired media is described in IEE (a), and encapsulation on IEEE wireless LANs in IEE (c).

One of the advantages of the EAP architecture is its flexibility. EAP is used to select a specific authentication mechanism, typically after the authenticator requests more information in order to determine the specific authentication method to be used. Rather than requiring the authenticator to be updated to support each new authentication method, EAP permits the use of a backend authentication server, which may implement some or all authentication methods, with the authenticator acting as a pass-through for some or all methods and peers(Aboba et al., 2004).

### B.4 Lightweight Directory Access Protocol (LDAP)

Lightweight Directory Access Protocol (LDAP for short) is a directory service for managing user account. It's an open Internet standard, produced by the Internet Engineering Task Force (IETF), the same body that gave the world TCP/IP as an alternative to the heavy weight DAP.(Howes & Smith, 1997) LDAP is a directory service, it is a simplified database, and should not be confused with traditional database. Typically, it does not



have the database mechanisms to support transactions.(Wang et al., 2008) Directories are flexible, secure, and can be personalized, another defining characteristic of a directory is that it supports Information Extensibility and Data Distribution(Howes et al., 2003).

Computer system requires the compatibility to locate certain types of information easily, efficiently, and quickly. As a directory service, LDAP is reasonably simple, but provide a wealthy of features, it consolidate existing services into a single directory that can be accessed by it's clients from various vendors, such as browsers, email clients, and authentication protocols(Carter, 2003; Howes & Smith, 1997).

In the 1980s, two separate standard bodies, International Telegraph and Telephone Consultative Committee, now know as International Telecommunication Union and International Organization for Standardization (ISO) started work on directory services for their own purpose. Eventually, the two independent directory specification efforts merged into one effort, and X.500 standard was approved in late 1988 then published in 1990. Like other open standards, LDAP have many implementations, from property software, to open source software such as openLDAPHowes & Smith (1997); Voglmaier (2003).

## B.5 ActiveDirectory

“Active Directory (AD) is a Microsoft network operating system (NOS) directory, build on top of Windows 2000 and Windows Server 2003”(Richards et al., 2006; Hunter, 2006). In 1997 Microsoft release Active Directory beta as the basis of their domain management function(Richards et al., 2006). The initial version was released with Windows 2000 and Windows Server 2003(Hunter, 2006). Active Directory proofed to be one of the most important new features in Windows NT 5.0. It greatly simplify the tasks involved in administering and managing large Windows NT networks, and it improves the user's interaction with networked resources.

The Active Directory is also the key underpinning that enables the improvements in distributed system security. All resources are stored and managed on the network,

it's easy for developers, administrators, and users to find and use. It provides a single, consistent, open set of interfaces for performing common administrative tasks, such as adding new users, managing printers, and locating resources throughout the distributed computing environment(Solomon, 1998).

The Active Directory data model has many concepts similar to X.500. It includes many other useful features such as group policies and delegation of authority that are useful in a diverse and distributed environment such as the university(Blezard & Marceau, 2002). The directory holds objects that represent various resources, which are described by attributes. The universe of objects that can be stored in the directory is defined in the schema. For each object class, the schema defines what attributes an instance of the class must have, what additional attributes it can have, and what object class can be a parent of the current object class. This directory structure has the following key features:

- Flexible hierarchical structure.
- Efficient multimaster replication.
- Granular security delegation.
- Extensible storage of new classes of objects and properties.
- Interoperability through Lightweight Directory Access Protocol (LDAP) version 3 support.
- Scalability to millions of objects per store.
- Integrated dynamic Domain Name System (DNS) server.
- A programmable class store.

Programmability and extensibility are significant capabilities of the Active Directory. Developers and administrators deal with a single set of directory service interfaces, regardless of the installed directory service(s). The programming interface, called the Active Directory Service Interfaces (ADSI), is accessible by any language. You can also

access the directory using the LDAP API. The LDAP C API, defined in RFC 1823, is a lower-level interface available to C programmers.

To establish if the functionality offered by Active Directory above NT domains is actually required we need to examine the functionality required in our environment for the support of Windows clients. Some simple questions need to be asked. These questions include the following:

- Is there anything that Microsoft Active Directory domain offers that is not readily available from Samba based Windows NT or (if supported in Samba.Latest) Active Directory domains?
- Is there anything that we need that Samba cannot provide to Windows clients?

One requirement that is clear is the availability of simple tools for use by systems administrators for the management of existing user accounts and resources. From experience it is the opinion of the author that the majority of regular sysadmin tasks should be performed by scripting rather than by hand. This is the best way to ensure the consistent application of settings, options and parameters to user and resource accounts. It is true that in certain cases, individual settings may need to be verified and for this a graphical tools is desirable. However, in many cases, the assertion or re-assertion of such settings using a pre-defined script tool is an equally sure guarantee that the required settings have been applied. The different opinions on this matter are somewhat analogous to the difference between a stateful and a stateless approach to system management. While one system administrator might prefer to check the settings applied to a user or resource and to do this with a graphical tool, another might ignore the current state and simply apply a pre-defined state using a configuration script.