CRESCENT+
A Secure Reliable Framework for Durable Composite Web Services Management

By
Sara Khurshid
NUST201362792MSEECS63013F

Supervisor
Dr. Shahzad Saleem
Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters in Information Security (MS IS)

In
School of Electrical Engineering and Computer Science,
National University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(July 2017)

Approval

It is certified that the contents and form of the thesis entitled "**CRESCENT+**" submitted by **Sara Khurshid** have been found satisfactory for the requirement of the degree.

Advisor: **Dr. Shahzad Saleem**
Signature: ____________________

Date: ____________________

Committee Member 1: **Dr. Muhammad Moazam Faraz**

Signature: ____________________
Date: ____________________

Committee Member 2: **Mr. Fahad Satti**

Signature: ____________________
Date: ____________________

Committee Member 3: **Mr. Ubaid-ur-Rehman**

Signature: ____________________
Date: ____________________

Dedication

Dedicated

to

My family and my friends who have always been my biggest support and to all my teachers since the first grade who are the real reason I am where I am today.

Certificate of Originality

I hereby declare that this submission titled **CRESCENT+: A Secure Reliable Framework for Durable Composite Web Services Management** is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics which has been acknowledged.

Author Name: Sara Khurshid
Signature:_____________

Acknowledgment

I am greatly thankful to Allah Almighty for this immense achievement in my academic life. For without Allahs blessings, I would not be here and celebrating this joyous moment. I am also thankful to my parents, who have been my greatest support since the first day of my school. I can never thank them enough for their love, kindness, tolerance and everything they have provided me with.
I have worked under the supervision of some great teachers, who have my immense respect for their guidance and support. I would like to thank Dr. Shahzad Saleem, Dr. Awais Shibli, Dr. Moazam Faraz, Mr. Fahad Satti and Mr. Ubaid ur Rehman for supervising my research work. I would specially like to thank Mam Rahat Masood, for her endless support and always being more than a mentor to me. Thank you maam. Also, I am very thankful to Dr. Islam Elgedway who remained a part of my research contributions and gave ample guidance. I feel privileged to have worked with such talented faculty. Working together with my colleagues and teachers has been an amazing experience for me. Specially, my KTH-lab fellows at SEECS, Majid Amjad Hussain, Obaid-ur-Rehman, Zohaib Shahid, Sadia Khalil and Yumna Ghazi have been a source of constant support for me. I am forever grateful to them for that. This journey had its ups and downs, but Ill never be able to forget it. Thankyou guys. And my friends who have always been there for my help whenever I asked or needed it, let it be moral or technical, Danish Azeem, Sehreen Jafry, Tanzeela Bilal, and Asad Ikram, I am very thankful to them and always will be.

Sara Khurshid

Abstract

Currently, a lot of service providers, provide the users with efficient atomic web services as per their requirement. But, owing to the increasing value of automation and complexity of requirements of users, there is a need to develop complex webs services and provide one stop shops for users. Integration/composition of two or more web services helps to reduce complexity for clients, thus increasing usability and providing better opportunities to attain complex functionalities in a simpler way. Creation and management of composite web services being a complex task in itself has not been sufficiently researched and discussed in literature so far. But, recently, it has become an active research topic in the field of web services. Many solutions have been proposed in this regard; however, they do not address the security of web services composition management. Furthermore, a lot of uncertainty aspects exist that could most likely cause hindrance in efficiency of composite web services delivery like possibility of any involved module to fail. Also, unexpected user demand raises can most likely become a reason of depletion of resources as well as can cause the service capacity and ultimately its performance to degrade. In addition to that there is a need to ensure security for all involved tasks and user/module interactions. Owing to these and many other security related challenges, this research focuses on these issues and provides a comprehensive secure framework for web service composition management. In order to carry out our research, we analyzed CRESCENT, which is a reliable framework for management of composite web services. The framework ensures reliable automated management for all composite web services life cycle tasks. We identified the security vulnerabilities in CRESCENT with intent to provide a valuable and feasible solution for the overwhelming security challenges of this framework. CRESCENT has been proven to be an effective and reliable composite web service management framework. Our solution includes addition of security mechanisms into the framework that will allow the users to avail desired web services in a more sophisticated way without violation of any of their security requirements. We implemented a secure session mechanism as proof of concept for the whole framework.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Chapter 1 aims to discuss the overview of composite web services that are the basic concept of this research. It also intends on providing an overall purpose of the research, together with the complete organization of this thesis document. The reason to conduct research in the domain of composite web services security is described in this section in detail.

1.1 Introduction

Currently, a lot of service providers, provide the users with efficient atomic web services as per their requirement. But, owing to the increasing value of automation and complexity of requirements of users in modern world, there is a need to develop complex webs services and provide one stop shops for users. Integration/composition of two or more web services helps to reduce complexity for clients, thus increasing usability and providing better opportunities to attain complex functionalities in a simpler way. This could lead to effective automation of business-to-business collaborations. Creation and management of composite web services being a complex task in itself has not been sufficiently researched and discussed in literature so far. But, recently, it has become an active research topic in the field of web services. Many solutions have been proposed in this regard; however, they do not address the need for a secure management framework for reliable composition of web services. Furthermore, a lot of uncertainty aspects exist that could most likely cause hindrance in efficiency of composite web services delivery like possibility of any involved module to fail. Also, unexpected user demand raises can most likely become a reason of depletion of resources as well as can cause the service capacity and ultimately its performance to degrade. In addition to that we need to ensure security for all involved tasks and interactions.

No web service composition management framework has been proposed so far. The mechanisms proposed till now do not focus on management and do not address security of individual as well as of the composed web services. Before we discuss our research workflow and our proposed solution, we will first provide a brief introduction to the domain of this research. This will help the readers to catch a glimpse of the domain and will aid them in comprehending the need for the solution that will be discussed and explained in the following chapters.

1.1.1 Composite Web services

Introduction

Web services in general could be defined in a variety of ways. But to understand their actual concept we can describe them as a group of loosely coupled and reusable software modules that perform a certain function. These functionalities are accessible over the internet. The results of these functions are submitted to users over the internet using standard technologies such as XML and protocols like SOAP. The web services are usually invoked using a Uniform Resource Locator (URL) specific to every web service. Composite web services are a more sophisticated form of web services which are basically a combination of two or more than two atomic web services. The web services are composed using various methods such as orchestration. Composite web services provide one stop shop for customers who require complex web services comprising of two or more web services at one single point of time.

Need for composite web services

Atomic web services are being extensively used to utilize various types of functionalities over the internet. The service providers create and provide services to users all over the world. The users are able to use them using a simple URL for any service they require. Although web services have created a revolution, still in the modern world they are now perceived as a bottle neck for organizations that need to use web services that are able to cope with their dynamically changing complex business needs and requirements. Such organizations cannot rely on a single web service as it may not satisfy changing system requirements in their dynamic systems. To cater this problem, composite web services are now being used. Composition of web service provides a inexpensive, effective, and efficient means for providing complex services through application integration over existing resources of organizations.

Limitations

Utilizing web services over the internet provided by third parties is now considered to be insecure in terms of many users security requirements. Users have grown more security savvy with passing time and increased awareness. Users are now stingier than ever regarding security features of anything they utilize over the network. They need to be aware of the providers credibility as well the privacy of the information they provide online. Security being the most desired nonfunctional requirement of the users all over the world is now one of the major concerns for software, application and framework developers. It is given a huge amount of importance when it comes to satisfying users requirements. Management frameworks for composite web services in specific that have been proposed so far offer no security whatsoever. Therefore, absence of effective security provisioning overall can be viewed as one of the limitations in usage of management frameworks for composite web services.

1.2 Motivation

Introducing security features into the dynamic processes like service composition is a big challenge nowadays. We have done a detailed survey regarding existing techniques proposed by various authors that cater to the provisioning of security features in their solutions. We carried out a comprehensive literature survey and a comparative analysis of existing/proposed techniques for overcoming security vulnerabilities with security mechanisms. The literature review has been discussed in detail in Chapter 2. We found that researchers have proposed many different security solutions till now based on requirements of users, service providers and even whole organizations. A holistic solution that provides a complete secure management framework for composite web services hasnt been proposed yet. In order to provide users and organizations with integrated web services for provision of secure sophisticated services, this issue has to be pondered upon. Our motivation is to provide a secure management framework for management of costumers SLA and service capacity, discovery of components, together with monitoring, coordination, cancellation and billing management. For the provision of such features together with a security aspect for ensuring confidentiality, integrity, availability, mutual authentication, authorization and non-repudiation, we propose CRESCENT+, a secure reliable framework for durable composite web services management. For examples, if a customer wants to shop online using the services of both a shop and a bank. Services of different service providers need to be integrated, in our case, one from bank and one from any shop, from where a customer needs to shop. Web service composers usually wish to utilize and integrate the shop and bank web services to provide users with a single point of access to customers that want to purchase items from a shop. For such integrated services, banks and other organizations will be required to collaborate.

1.3 Problem Statement

None of the solutions proposed so far is holistic in their approach and does not cover all the security aspects required for a secure management framework. Therefore security of management frameworks for composite web services has not been discussed till now. This research problem is still in its infancy and mature solutions are not available. Therefore, our research focuses on providing a secure architecture for CRESCENT which is a durable and reliable management framework for web service composition.

1.4 Aim and Objective

Our aim for this project is firstly to carry out an extensive and sound literature survey on vulnerabilities and challenges in web service composition process related to security aspects. And more particularly, carrying out the security vulnerability study related to CRESCENT, the reliable management framework for efficient service composition. To achieve our objective, there is a need to propose a secure management framework for web service composition, which will help customers and service providers to securely avail and provide online services respectively without facing any threat to their personal information or data. Customers will be able to avail set of multiple services from a single access point.

1.5 Research Contributions

CRESCENT, a Reliable Framework for Durable Composite Web Services Management, has been proven to be an effective and reliable composite web service management framework. In order to carry out our research, we will analyze CRESCENT. This framework ensures reliable automated management for all composite web services life cycle tasks. The security vulnerabilities in CRESCENT have not been discussed so far. This research project is intended to provide a valuable and feasible solution for the overwhelming

security challenges of this framework.
This research project will provide:

1. A thorough and broad literature survey of existing security solutions for composite webs services to analyze security research challenges being faced in implementation and management of secure composite web services on internet. Chapter 2 will discuss in detail the solutions for secure composition of web services. It will also include the analysis for analyzing their shortcomings.

2. Detailed analysis of security requirements of CRESCENT using STRIDE methodology. The methodology will help to analyze vulnerabilities of CRESCENT and also will provide a guide map regarding the features to be incorporated that will provide security to the framework. The detailed vulnerability assessment will be discussed in Chapter 3 of this thesis document.

3. The above two will lead to proposing a reliable and secure composite web services management framework named CRESCECNT+. Our solution will involve incorporation of security mechanisms into the framework. The users will be able to use desired web services in a more sophisticated way without violation of any of their security requirements.

Major advantages of the proposed framework are:

- Framework will promote importance of security in management of composite web services. It will highly encourage service providers all over the internet to carry out revolutionary research in the security aspect of composite webs services management frameworks, as no significant work has been done in this regard, so it is a very talked-about research topic in the field of cloud computing, e-shopping, e-banking, web services and composite web services.

- Provisioning of secure reliable services at one point to customers.

- Service Providers can also then mange the secure composition of web services more effectively and efficiently.

For our research, we have used a combination of various research methods and followed the resulting methodology for addressing the domain of secure management of composite web services. The methodology comprises of a systematic and critical approach for analysis of existing solutions proposed in regard to composite web services security. Also, we have thoroughly

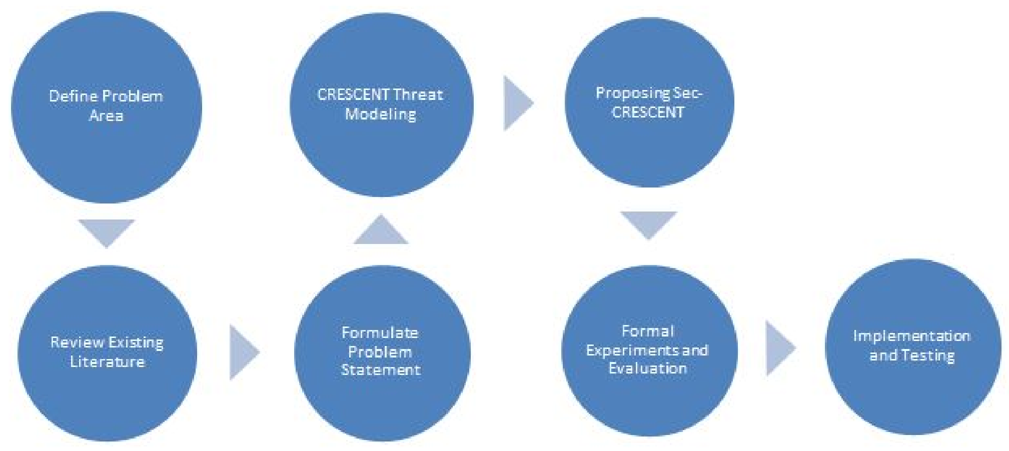

Define Problem Area
CRESCENT Threat Modeling
Proposing Sec-CRESCENT
Review Existing Literature
Formulate Problem Statement
Formal Experiments and Evaluation
Implementation and Testing

Figure 1.1: Research Methodology of the Research Work

analyzed CRESCENT, to formulate a secure composite web services management framework through threat modeling and designing attack trees for fund vulnerabilities.
The methodology followed in this research is shown in the following Figure 1.1

1. **Define Problem Area:**
   Our research incudes in depth survey on current technologies of web services and their composition. Based on the survey, we figured out a number of requirements of advanced field of composite web services, most important being the security in this regard.

2. **Review Existing Literature:**
   In our research, we have analyzed existing solutions and identified their shortcoming through comprehensive analysis

3. **Formulate Problem Statement:**
   Through detailed analysis in the previous phase, we were able to identify the need of a holistic web service composition management framework that still fails to exit.

4. **Threat Modeling of CRESCENT:**
   As a next step, we chose CRESCENT based on it holistic approach regarding provisioning of a complete framework for reliable web service composition. It catered for significant number of nonfunctional requirement of user in an efficient way. We used STRIDE as the threat modeling technique for identifying security issues with the framework

to specifically understand the design of the solution that we were to propose in the next step.

5. **Proposing CRESCENT+:**
   After identifying the vulnerabilities, we added security modules to cater for the issues and propose a secure framework as a result, CRESCENT+.

6. **Formal Experiments and Evaluation:**
   We then performed a number of experiments on the solution proposed and found out from the results that, the solution does enhance the security as well as reliability of the original framework.

7. **Project Implementation and Validation:**
   Later, as a next step of our research, we implemented one of the major starting point of our framework as a solution on small scale and validated it.

1.6 Thesis Organization

The thesis document has been divided into well-organized chapters that will be easy to follow and read through. Every chapter pertains to important aspects of the research work.

Chapter 1 ***Introduction and Motivation***
In this chapter, we have discussed an overview of basic concept of our research which is composite web services. This also provides the purpose, aim and objectives of the research work that was carried out.

Chapter 2 ***Background and Literature Review***
This chapter includes a detailed literature survey of the different techniques, models and solutions that have been proposed so far for provisioning of secure composite web services to users. This also includes an analysis of the mentioned solutions and highlights the aspects that are deemed important from security perspective.

Chapter 3 ***CRESCENT***
This chapter includes introduction to an existing composite web services management framework that has been selected for performing security requirement analysis to propose a thorough and secure composite management framework by identifying its security vulnerabilities. We have also discussed

the technique that we had used to identify various security vulnerabilities of the CRESCENT framework. We have used STRIDE, a model for thorough vulnerability assessment of the framework for identifying a number of threats categories. These categories were then visualized using attack trees that provided insight regarding possible exploitation of the framework by malicious entities.

Chapter 4 ***Design nd Evaluation of Proposed Solution***
Chapter 5 includes complete description of the solution that has been proposed, CRESCENT+. The security issues analyzed in the previous chapter through threat modeling and their visualization been done through attack trees, were catered for using various security modules. Each security module handles a different security feature employing a specific security mechanism. Execution flow of each security module with respect to CRESCENT has been described. Later, we have also added the whole workflow of the CRESCENT+ framework for service delivery.After the description of workflow, we have evaluated the addition of security modules with respect to the previously mentioned threats and vulnerabilities of the framework. The functionalities offered by security modules effectively cater to the identified security requirements of the framework.

Chapter 5 ***Design and Implementation***
This chapter includes implementation details of the research work together with the discussion regarding benefits achieved. The implementation presents proof of concept for one of the security aspects for the CRESCENT framework, that is user/module registration (Authentication),logging in and secure service utilization. The technologies used have been discussed.

Chapter 6 ***Conclusion***
In chapter 6, this thesis document is concluded by stating the overall problem, objectives achieved and contributions made through this research work and possible future directions that could be considered for any future work.

Chapter 2

Background and Literature Review

Chapter 2 discusses the literature survey carried out for identifying the research problem as well as the security features that need to be adopted by current solutions for effective and secure web service composition. The chapter discusses in details the solutions that have already been proposed. It also provides an analysis of these solutions together with their comparison that highlights their limitations and shortcomings with reference to the security features defined at the start of the chapter.

2.1 Background

The increase in number of web services, as the easily accessible programs over the internet and the emergence of semantic web, both have together enabled assembling of complex web services and their delivery over the internet. A durable composite web service can be regarded as a web service that helps to realize a process that involves a number of component web services for various functionalities. Such web services have long life time and customers are likely to trust them more for their functions. Any service that is created should always fulfill customers SLAs no matter what. Although ensuring SLAs when the composite web service is using unreliable components is a very challenging task. As the components are likely to perform unexpectedly and give unexpected results any time during their usage. Also, this may result into component failure that may cause the failure of composite web service itself. Such failures may be caused due to physical problems or else SLA violations, or computational errors.

Literature review reveals that, failure of a composite web service might be caused due to the Byzantine failures of its components web services (Lamport et al., 1982). Similarly, whole delivery process could be compromised in case of failure of any sub module specifically ones related to service management. Also, the demand of customers for composite web services may exceed above defined limits and lead to depletion of resources (Ghosh and Naik, 2012), eventually causing customers to wait longer and eventually losing their trust in durability and efficiency of the service provider. All these factors are involved in posing business risks of high level. The composite web service providers therefore carefully maintain and manage their provided services and ensure that the terms mentioned in SLAs of customers, are met properly and as per their desire.
Presently, service providers try to manage such risks manually. For example, in case a component web service is affected or fails, the service providers replace them with alternates or enhance the capacity of their system based on the highest/worst possible scenario of user demands. Managing such service requirements and handling failure possibilities and evading them is definitely a costly and time consuming process. In this light, we present that, we need to use a service management system to handle such business risks. This system should be able to satisfy customers' SLAs and provides support for delivery of durable composite web services through provisioning of following features:

1. **Byzantine Fault Tolerance BFT Delivery** even in case of component failures
2. **Automated Capacity Management** in case of demands spikes during various time frames
3. **Automated Coordination** between different components for reliability and correctness of results and Execution.
4. **Automated Task Recovery** from possible failures of invoked components as well as modules
5. **Automated SLA Management** and workflow customization for customers
6. **Automated Dynamic Adaptive Composition** as per users requirements and SLAs without any violation.
7. **Automated Components Discovery** for sake of usage in composition plans for creation of composite web services

8. **Automated Cancellation Management** of users, components as well as billing

9. **Billing Management** through automatic bill creation, based on policies in SLAs and agreements

10. **Automated Security Management** to help avoid threats like confidentiality, integrity, privacy, availability, authentication, authorization etc. which are a major concern to deal with. Especially, existing standards such as WS-Security do not address service composition in particular and mainly focused on atomic web services

Unfortunately, we were unable to find any existing work that proposes/ presents a durable delivery system for composite web services that fulfils all the above mentioned mandatory requirements. The existing work is focusing on Byzantine Fault tolerance for atomic web services such as (Zhao, 2007a) (Pallemulle et al., 2008) , (Castro et al., 2003) , (Merideth et al., 2005). However, there exists some work that addresses the issue of fault tolerance in composite web services management such as (Zhao and Zhang, 2008) , (She et al., 2008) , (Liu et al., 2010) Work in (Zhao and Zhang, 2008) focuses on coordination Byzantine fault tolerance between the components but ignored individual components fault tolerance at all. Also, work (She et al., 2008) , (Liu et al., 2010) focused just on components fault tolerance without supporting Byzantine faults tolerance. Thus, we can conclude that no existing work supports Byzantine (or even normal) fault tolerance for both components and coordination modules, which we consider to be an important requirement for reliable composite web services delivery process. Therefore, in (Elgedawy, 2014) author proposed CRESCENT; a BFT management framework for durable composite web services. CRESCENT has all the features discussed above; except the security management requirement. Hence, it lacks the security considerations such as authentication, authorization, availability, confidentiality, integrity and non-repudiation. There is need for incorporating security requirements in design and implementation of the framework for enhancing its usability in security critical environment, such as encryption, for secure access of SLAs and workflows for creation of reliable composition plans for realizing the web service requests. The storage pools containing the SLAs, workflows and components could be accessed and altered by any unauthorized user. Security mechanisms such as hashes and encryption are required, to ensure no such violation occurs. Thus, the proposed framework CRESCENT+ provides a secure and effective way to create and manage composite web services in dynamic internet systems such as cloud. This ensures effective provisioning of services to the customers.

2.2 Literature review

The field of web engineering has grown a lot in the past decade. It gained wide acceptance as a platform for sharing data. And it therefore led the scientists to develop high quality web applications using various sophisticated tools and scientific principles increasing number of researchers are focusing their research to web technologies and their usage to develop, deploy and maintain web applications. Usually, a sound process is required for the development of a web application which has been a great motivation for developers for finding and recreating new methodologies and tools for specific purposes.
Web service is a promising distributed service-oriented technology that provides distributed services on request and effective automation of business to business associations. It can be bound and interactively requested for service over the internet. Web services are built over XML standards like Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description, Discovery and Integration (UDDI). Although a single web service can be of value, combining web services into composite web service can bring more value and service. Components involved in web service composition remain physically separated and it is determined which component to call and in what order. This Service Oriented Architecture (SOA) technology allows automated interactions of distributed diverse kinds of applications and is a substitute for hard coding such applications. For composition of web services, technologies or languages like BPEL4WS, BPML, and WSBPEL have been proposed. BPEL is evolving as a declarative standard and describes business processes on the basis of interacting web services. BPEL4WS used for composition f web services, that is, it generates complex processes by creating and wiring together different operations which can call web services and modify or use data, or stop the process. Despite many benefits provided by web service composition, security threats like confidentiality, integrity, privacy, availability, authentication, authorization etc. are a concern to deal with. To address SOA security requirements, several specifications are there: WS-Security, WS-Addressing, WS-Policy, SAML, WS-Security Policy etc. The aim of this research project is to carry out an extensive literature survey related to security aspect in web service composition and management, its potential security requirements, vulnerabilities, threats it faces, key challenges and mechanism to make it secure for providing a secure solution for composing web services. Introducing security features into the dynamic processes like service composition is one of the challenging tasks nowadays. Researchers have proposed many different security solutions till now based on requirements of users, service providers and even whole organizations.

This literature survey includes various approaches that have been proposed. One of them focuses on formulation of security policy for composite web services. This work by far only focuses on ensuring consistency between the atomic and composite web service policies. No security aspects have been considered. Some work has also been done related to access control models, for securing the sensitive data of users in cross-domain services composition. An architectural framework for academic institutions has been proposed with an aim to provide secure composite web services using multi-level security concept. A number of security features have been provided, but the framework primarily focuses on authorization and authentication with no attention to secure communication, or integrity of data at rest and in transit, thus vulnerable to a number of threats. Another framework has been presented to ensure secure decentralized execution of composite web services. The framework can provably provide certain security features but lacks important features such as authorization and non-repudiation. In addition to these frameworks, security models addressing security issues of both customers and service providers have been proposed. None of the solutions is holistic in their approach and does not cover all the security aspects required for a secure management framework. A comparative analysis of existing/proposed techniques for overcoming security vulnerabilities with security mechanisms have been carried out to propose a secure architecture for CRESCENT framework. It has been found that no management frameworks for composite web services have been proposed so far. Therefore security of management frameworks for composite web services has not been discussed till now. This research problem is still in its infancy and mature solutions are not available.
In the following sub section of this chapter, we present a survey of different architecture frameworks proposed by different researchers in the domain. The purpose of this survey is to state our findings and summarize the security features being provided by proposed solutions of different researchers. The literature we surveyed addressed different problems like rules and security constraints to consider when composing web services, making the composition of web services privacy aware, making use of aspects while composition, providing QoS while composition, providing multi-level security.

2.2.1 Security of Composite Web Services

Although, considerable amount of research work has already been carried out in enhancing the security of composite web services, no holistic solution has been proposed or implemented yet to improve the security of composite web service management. Although there exists voluminous literature on improving the web services quality and also some work has been carried out in their security but research is still to be performed in providing reliable and secure web service management frameworks. Below, we have discussed and analysed the security of composite web services and the solutions that have been presented so far in the respected domain. Table 2.1 shows the findings of our analysis based on important security requirements for composite web services.

There are no clearly established rules for composition of policies in composite as well as atomic web services without inconsistencies. Work has been done to check policy inconsistencies (Tziviskou and Di Nitto, 2007) , (Lee et al., 2006) but still they do not address the need of policy composition rules for arbitrary composite processes. Fumiko et al. (Satoh and Tokuda, 2011) propose a framework that provides a semi-automatic method for formulating a security policy of the composite service by defining the process-independent policy composition rules for ensuring consistency among composite and atomic policies. The rules help in defining composite policy for Message Protection and Access Control. The major advantage of the approach is that policy consistency rule is independent of any particular composite processes but majorly focuses on policy composition for composite web services and no other security feature. Numbers of access control models have been proposed for atomic web services (Coetzee and Eloff, 2004) , (Goettelmann et al., 2013). Yan et al. (feng YAN et al., 2013)propose the mechanism, privacy-aware role based access control model for Web services composition (WSC-PRBAC), for securing the sensitive data of users in cross-domain services composition. The mechanism uses the existing RBAC systems with added privacy related entity, thus specializing in privacy enforcement. The paper only focuses on privacy provision and no other security feature in particular. Sathiaseelan (Sathiaseelan, 2013) proposed an architectural framework for academic institutions with an aim to provide secure composite web services using multi-level security concept. Number of security features such as authentication, data confidentiality, data integrity and authorization is provided using multi-level security. The framework primarily focuses on authorization and authentication with no attention to be paid on secure communication. The framework claims to provide multilevel security however, significantly lacks in covering the original concept of multi-level security. Also proposed framework is just a concept and its full-fledge implementation is completely missing. Additionally, the framework also fails to support integrity and confidentiality of data at rest and in-transit. Akin to Sathiaseelan (Sathiaseelan, 2013), Joachim et al. (Biskup et al., 2007) also presented a framework in order to ensure secure decentralized execution of composite web services. The framework is extensive enough to support fundamental security features such as authentication, message protection and integrity but lacks to fulfil requirements of authorization and non-repudiation. Wei She et al. (She et al., 2008) extended the basic security model by introducing the delegation and pass on policies in composite web services. A secure interaction and information flow control is established via access control. Same as others, this model is limited only to access control and is not intended to involve security features such as authentication, confidentiality and secure communication. A service supervision system has been introduced by Masahiro et al. (Tanaka et al., 2009) that controls the execution of composite web services in an open environment. The security of framework, however, did not highlight access control issues as well as the security policies of the users have not been taken into account. Charfi et al. (Charfi and Mezini, 2005) in their work regarding securing of BPEL, also discussed the security policies of the interacting partners which are duly checked at deployment time. The mechanism secures the BPEL compositions using WS-Security and WS-Policy and discusses the BPEL activities (invoke, receive and reply) in the context of confidentiality, authorization, authentication and integrity along with the integration of security-related properties in BPEL specification. Although, the above framework provides a number of security features, but still fails to address the fault tolerance of its components framework as well as of composite web services. Carminati at al. (Carminati et al., 2005) discusses the security concerns of composite web services by considering both web services provider (web services) and requestor. A security model based on web services standards, has been proposed to represent security constraints. The proposed model is formulated into brokered architecture whose one component is responsible for matching the compatibility of web services with security constraints. This work has been further extended by Cariminati et al. in (Carminati et al., 2006), where they presented Security Vocabulary/Ontology to model security information related to web services after the investigation of security constraints in webs service composition. They used Web Ontology Language (OWL) (van Harmelen et al., 2003) to describe the security constraints and capabilities of web services. WS-Agreement (Andrieux et al., 2007) have been particularly used to model the security constraints and capabilities of web services. These approaches however do not discuss management of web service composition or the byzantine fault tolerance. Biskup (Biskup et al., 2007) et al. propose a framework for the secure execution of composite web services. The framework consists of different layers, which ensure the authenticity, confidentiality and integrity requirements. The access control is provided by restricting the web services to access the information only on need to know basis. Confidentiality and authenticity is enforced through symmetric and asymmetric cryptography and digital signatures. This framework however, does not address the creation and management of customers' SLAs which are important for realizing the appropriate service, as per customer's request. Goettelmann et al. (Goettelmann et al., 2013) proposed an approach to securely deploy business processes in hybrid Cloud environment in order to ensure protection of data. The proposed approach uses the partitioning technique, fragmenting the centralized process, to fulfil security requirement and communication cost optimization. Karimi et al. (Karimi and Babamir, 2010) focus on security and QoS issues in web services composition. They proposed a brokered architecture for composition of web services based on the specified security related constraints. Two major non-functional properties, security constraints and QoS attributes, have been discussed that have a great impact on web service composition. The main purpose of proposing this mechanism is to help brokers in composing web services according to the security requirements of web service requestors as well as the providers, after filtering them by their QoS properties. In (Hutter and Volkamer, 2006), Hutter et al. have added a security type' with sensitive data being used by services to ensure access control and to prevent unauthorized access by any service. This increases the confidentiality as well as integrity of the personal information. However, the pro-active agents dynamically compute the data for performing the task of delivery of service at run time. The security policy of customer helps the web service and guides it on how to use the provided data when interacting with other web services. It is ensured that the execution of request doesn't violate any security policy. The proposed approach results in ensuring the confidentiality and integrity of personal customer information over the internet. The Souza et al. in (Souza et al., 2009) identify a collection of security requirements of service composition and present an approach called Sec-MoSC to incorporate the requirements into security mechanisms. The important security requirements include data encryption, authentication of web services, event logging etc. The major advantage of this approach is that it is easily extensible. Existing solutions on the security of service compositions focus on a particular aspect such as incorporation of security requirements into business processes (Coetzee and Eloff, 2004) , (Koshutanski and Massacci, 2004) , (Joshi et al., 2001) or enforcement of security during execution (Coetzee and Eloff, 2004) , (GRANT and INTELLECTUAL, 2002). There are also few techniques that defines the authorization architectures (Bhatti et al., 2005) , (Koshutanski and Massacci, 2004), whereas few approaches deals with the capability-based access control (Charfi and Mezini, 2005) for composite web services. Other proposals in the area of composite web services address the matching of security constraints/requirements of composed web service environment with the security capability of individual web service.

However, none of them deals with the problem addressed by this research, that is, formulating a threat model for composite web services management system; CRESCENT; and providing a framework which deals with the fundamental security features of authenticity, confidentiality, authorization and secure execution of all components of system. There is also little work in the identification of threats and requirements for composite web services, yet research and development needs to be performed for web services management systems and frameworks.

Research Findings and Analysis :

We have analyzed various techniques that have been proposed for providing security to composite web services. The objective of our analysis is to compare these techniques based on the criterion of core security features, shown in Table 2.1. This comparison can assist service providers in selecting the most appropriate technique for security of their composite webs services based on their security requirements. We have found that most of the proposed security solutions do not offer support to all the essential security features that composite web services require for secure execution and the ones that do, have their own weaknesses. None of the discussed techniques heuristically covers all the security features. Table 2.1 provides a brief overview of the reviewed approaches and important security features they offer. The analysis of differences in the techniques and offered security features is presented in the following table, where, "Yes" depicts the feature that a technique provides while "No" shows the unavailability of that feature in a particular technique. (Carminati et al., 2005) discussed above is not compared with other architectures because it does not provide security rather focuses on security constraints which are considered while composing web services.

Table 2.1: Comparison of different techniques that provide security features for web service composition

Security Feature	(Satoh and Tokuda, 2011)	(feng YAN et al., 2013)	(Sathiaseelan, 2013)	(Bhatti et al., 2005)	(Charfi and Mezini, 2005)	(Biskup et al., 2007)	(Goettelmann et al., 2013)	(Karimi and Babamir, 2010)	(Hutter and Volkamer, 2006)	(Souza et al., 2009)	(She et al., 2008)
Authentication	✔	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗
Authorization	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
Availability	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
Integrity	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✗
Confidentiality	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗
Trust	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Privacy	✗	✓	✗	✗	✗	✗	✗	✓	✓	✓	✗
Non repudiation	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗

Chapter 3

CRESCENT

In this chapter we have discussed CRESCENT framework that we have chosen for our research. The functionalities are discussed in detail. We have also discussed the reason for choosing CRESCENT over other frameworks based on the features that CRESCENT provides for reliable and efficient composite web service management. Later, we have identified a number of security requirements of CRESCENT as it does not provide any security feature together with reliable service delivery.

3.1 Introduction:

Customers and composite service providers together define the user specific SLAs as a part of the contract of service, which includes different contractual requirements such as the required workflows, pricing, availability, actions to be performed when violations occur, endured penalties, etc. (Dan et al., 2004) , (Yu et al., 2008) , (Charfi et al., 2008). Such service contract is observed by one or more of the involved parties, and in some cases services from third-parties are also taken for this purpose. CRESCENT enables composite web services providers to automatically deliver their services without violating the required SLAs. CRESCENT has many unique features that differentiate it from existing solutions such as:

- In order to provide differentiated levels of service for customers, various types of workflows as required are defined and supported by CRESCENT.
- It also supports flexible and individualized automated SLA management, as SLAs are defined in a machine-understandable format. This is accomplished by extending the Web Service Level Agreement (WSLA)

language (Keller and Ludwig, 2003) which was initially aimed at atomic services.

- It dynamically realizes the required workflows using serial and/or parallel provisioning plans by finding the suitable component services that realize the required workflows via an adaptive composer module that finds the best composition plan for the SLA. This is done with the help of a service discovery module and a capacity planner module. Once the composition plan is finalized, the dispatcher module invokes the chosen components according to the required workflows. SLA and component monitors check for SLA violations, if such violations are found, the dispatcher automatically stops the execution, and resubmits the requests to the adaptive composer to look for alternate composition plans.

- It ensures Byzantine fault tolerance for delivery of composite web service by combining two protocols named quorum-based and state-machine-based BFT protocols. The quorum based BFT protocol ensures reliability of components execution, while the state-machine-based BFT protocol ensures reliability of delivery modules. CRESCENT ensures components redundancy via components parallel provisioning unlike many other approaches that do it via component replication. That for every task in the required workflows, a group of components will be invoked in parallel to realize the task. We define such a group of parallel components as a component cluster.

- It dynamically creates multiple component clusters for the same task when the demand over the composite web service is high, to avoid performance degradation.

- Dynamic composite web service capacity is managed by combining two approaches namely predictive and provisioning, in order to handle delivery requirements at different time-scale. This is done using a component pool that contains all verified and testing components. CRESCENT always ensures having enough working components in the pool using the discovery module. The components are intermittently checked and tested to make sure that they are valid.

3.1.1 CRESCENT Modules:

Figure 3.1 depicts the main modules of the CRESCENT framework and their interactions. More details about CRESCENT could be found in (Elgedawy, 2014), however we summarize its main modules as follows:

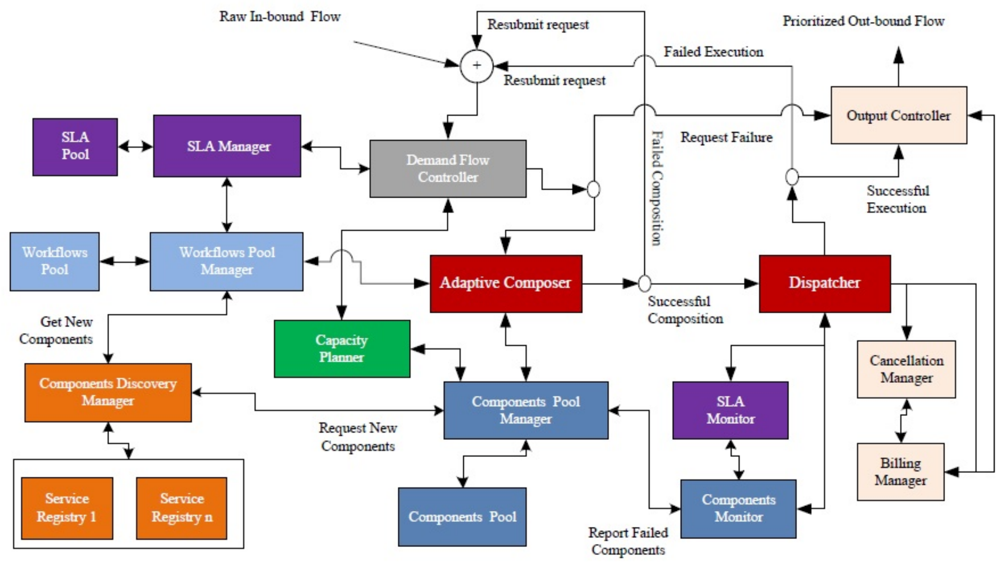


Figure 3.1: The Architecture of CRESCENT - A BFT Framework for Composite Web services Management

SLA Manager:

SLA manager communicates with customers and assist them to create their required SLAs, which could be made from scratch or else previously specified and stored SLAs could also be used. The customer is just required to add a reference to specific existing SLA in the submitted request. The manager further associates the SLAs with required workflows as per their priorities.

SLA Pool:

SLA pool contains all of the created SLAs for the customers. Customers can always choose SLAs of their own choice from already existing ones or else create new variants. This facility helps the customers by simplifying the request submitting process for the composite web service, as the customers could just simply refer to unique SLA ID of the SLA they want to add the reference to in their request. Thus, customers are not required to provide same SLAs over and over again for any service request.

Demand Flow Controller (DFC):

This module is in charge of directly communicating and receiving service requests from the customers. It also keeps track of the resubmitted requests in case of failures. As soon as failure threshold is reached for a certain request, a new next in priority workflow is associated with the request. Also, a failure count is maintained. This count then helps in creation of demand statistics that are later provided to capacity planner. If a all workflows are assigned and tried and failed, then DFC reports request failure. DFC receives the requests and prioritizes them based on SLAs and associates each request with the workflow of highest priority from the workflow pool. These requests are then forwarded to Adaptive Composer for searching suitable components for realizing the workflow.

Workflows Pool:

Workflow pool contains all the workflow that are creates by service providers against the services to be provided to the customers. Workflow pool manager manages the workflow pool and adds/deletes/updates a workflow whenever contacted to do so.

Adaptive Composer (AC):

The requests received from DFC are processed by Adaptive composer in a prioritized FIFO manner. Against every request, it creates a composition plan. The plan is then submitted to Dispatcher in the form of a BPEL script. The components are found and retrieved from components pool, for which components pool manager is contacted. If the required components for the plan are not found in the components pool, workflow of next in line priority is chosen for realization. The components are chosen based on pre-defined algorithm such as round robin, most recent, lest cost, etc. The logs of adopted composition plans are being maintained. In case a cancellation request is issued by the user, the log f specific plan is forwarded to dispatcher to issue its cancellation orders.

Components Pool:

It contains all the components that are discovered by Components Discovery Manager for sake of composition plan realization. Components are sorted and numbered based on workflows associated. This pool is managed by Components Pool Manager and therefore can update add/delete components as per requirements and demand. The manager is also capable of deleting any component that is found violating any customer SLA. If components are not enough at any time, discovery request for new components is send to Components Discovery Manager.

Capacity Planner:

This module receives demand statistics from DFC and ensures that capacity of framework is up to date and framework has enough components in components pool to handle incoming customer requests. Based on received statistics, it runs different forecast predication algorithms to figure out the required capacity to ensure appropriate number of components is always present in the pool. IF more components are needed, CDM is contacted.

Components Discovery Manager:

To realize tasks specified by workflows, CDM is responsible for finding new components from various internal and external registries. If components are not found, then components of next appropriate workflow in line are searched.

Components Monitor:

Based on defined business rules, components are constantly monitored on regular and on-demand basis. All information related to workflows is provided to Components Pool Manager to update the information of the components in the pool.

Dispatcher:

It is the orchestrator, coordinator, and initiator for the composition plan execution. It executes the composition plan provided in the form of BPEL script. In case of failed execution, request for service is resubmitted, this time framework checks for next best workflow and components. In case of cancellation, dispatcher sends request to cancellation manager and billing manager to ensure proper cancellation and compensation actions.

Cancellation Manager:

It is responsible for cancellation of resubmitted requests in case a customer desires to cancel the request. Also, proper billing adjustments are also ensured.

Billing Manager:

It is responsible for generating users bills and make sure payments have been made according to their SLAs and other important constraints. It is also responsible for paying the component providers of the services according to their SLAs. To know which components have been consumed, it communicates with Dispatcher and cancellation manager for updated information.

Output Controller:

It is responsible for providing the customers with the generated composite service responses. It also prioritizes responses based on the customers SLAs.

3.2 CRESCENT against Existing Approaches:

The major feature provided by CRESCENT is Byzantine fault tolerance. Many solutions exist that provide BFT one way or another but lack in delivering accompanying features for efficient and reliable service composition. Some of the research work (such as BFT-WS (Zhao, 2007b), PERPETUAL (Pallemulle et al., 2008), BASE (Castro et al., 2003), and THEMA (Merideth et al., 2005)) focuses on providing BFT for only atomic web services while others, that are very few (such as (Zhao, 2007a) (Onditi et al., 2008), and FACTS (Liu et al., 2010) discuss problems with respect to composite web services.
In general terms, all of the mentioned approaches vary in various aspects such as design, model and component replication management. It is important to ensure that components of the framework are replicated often to provide constant support of dynamic web service composition. The BFT approaches used for atomic web services cannot be utilized for composite web services as web service composition requires other major features such as coordination, recovery, monitoring, logging and isolation of faults and errors.
In contrast to previously proposed solutions CRESCENT provides a number of advantages which include isolation of composite service logic from its execution, as well as efficient management of errors and faults. It also provides support for request resubmission in case of failures. Other differences are shown in Table 3.1 that shows a comparison between CRESCENT and major existing approaches (Elgedawy, 2014).

Out of all discussed solution, no approach fulfils the requirements for reliable and efficient delivery of composite web services. However, only CRESCENT managed to fulfill all the requirements except security. This is why, we are extending CRESCENT in this research work and propose CRESCENT+ framework, to overcome such problem.

3.3 Security Requirements for CRESCENT:

CRESCENT is a purely management framework for composite web services that has not highlighted security features such as confidentiality, integrity, authentication, authorization etc. We have identified several security requirements for securing the data, resources and services that are being provided by the framework. These security requirements worked as a drive for the solution that we have presented in this thesis work.

3.3.1 Mutual authentication:

The modules of the framework need to mutually authenticate with each other during their interactions. This is required so that it is ensured that only authenticated modules could interact and access the critical or sensitive information from the relevant modules. For-example, the Workflow Pool Manager needs to be authenticated before performing actions such as Workflow Addition or Deletion from the Pool. Also, the user making request must be authenticated against his specific SLA at the time of request analysis by the Demand Flow Controller or the SLA Manager. This mutual authentication will help in preventing spoofing and masquerading attacks on framework modules as well as customers.

3.3.2 Confidentiality:

Confidentiality involves securing data at rest as well as the data being shared during the communications. The data being exchanged among modules as well as between the customer and the system must be secured and disclosed only to the modules that have to perform certain functions in delivering the service. Customers expect guarantees with respect to security of their data. If the modules like Adaptive Composer, are hosted on different servers, then the messages exchanged between modules over the network should be secure. Secure data communication ensures protection against external attempts of spoofing of the critical information regarding service composition and delivery. In case of shared server, no such attempt of securing inter module communication is required. Similarly, data at rest such as SLAs in SLA Pool, workflows in Workflow Pool need to be secured using encryption mechanisms for preventing unauthorized or illegal access to critical resources.

3.3.3 Authorization:

Service providers that are responsible for providing the composite web services should make sure that only concerned users and modules are able to access the relevant data being stored in pools i.e. Components Pool, Workflow Pool and SLA Pool. The access control mechanisms need to be used for ensuring authorized access to data.

3.3.4 Security Features Customization:

Customers must be allowed to specify security constraints of their choice at the time of SLA creation, which are later checked at various points in delivery mechanism for preventing violation at all stages. The security requirements imposed by the SLAs must be satisfied when the composite web service is being delivered by the service provider.

3.3.5 Data Integrity:

The data stored in the system i.e. SLAs, Workflows and Components is prone to misuse or alteration by unauthorized users so it must be checked against corruption at the time of Workflow or Composition Plan creation. These resources are managed by corresponding manager modules in the framework. Absence of authentication prior to any managing activity, may lead to alteration of the components, workflows or SLAs within the pools, by an unauthorized entity. Data therefore needs to be secured using either encryption or hashing techniques, such as SHA or MD5.

3.3.6 Non repudiation:

The modules need to be monitored for efficient service delivery. For that, it must be ensured that an action or a request was actually made by a specific user or an authorized module. For example, the service consumer that makes the request for a service at any time could be held accountable in case, he denies sending the request. Similarly, the addition and deletion of components from Components Pool by the Component Pool Manager needs to be tracked for auditing purposes in case of failures or unavailability of components in the Components Pool at the time of service composition.

3.3.7 Components Security Capability Monitoring:

The components discovered and stored for creating workflows against requested services need to be certified by a trusted third party and authenticated to ensure their reliable and secure behavior. In case of any alteration in component behavior, it must be certified again or else replaced by any other component with the similar functionality.

3.3.8 Auditing:

All the operations being performed by the modules for realizing a service must be logged for future reference to specific workflow or a composite web service. This feature can also help in evidence provision in case of any malicious attempt by an adversary. For example, if a module gets compromised i.e., module spoofing, leading to occurrence of some malicious activity by a module that violates any of the SLAs or security constraints. Such activities if properly logged, can help gather evidence against the adversary as well as keep a check on system activities on the fly, preventing serious damage due to negligence or ignorance.

3.3.9 Availability:

The service composition framework requires constant availability of CRESCENT modules, for example, for workflow and composition plan creation. Demand Flow Controller statistically calculates the demands of users and conveys them to the Capacity Planner. The algorithms must be authenticated to ensure minimum calculation and estimation errors, which could possibly result in lack of availability of the system at critical times. This feature will help in minimizing the failures reported due to component unavailability that tarnish the repute of the service provider.
Security requirements having being defined for CRESCENT framework, we will now look at how and at what entry points, the framework is vulnerable to security breaches or threats, using threat modeling.

3.4 Threat Modeling of CRESCENT:

Security features at the time of system development are not usually given prime importance and are treated as non-functional requirements. Whenever a certain system is developed, security features are generally incorporated during the design phase and security is provided as a built-in feature. Or else, these features are not addressed up front at the design phase but are later weaved into the system i.e., retrofitted. The systems then, prior to deployment, are tested from security perspective using attack data and corresponding test cases. In our case, we have presented a security solution for existing management framework, CRESCENT. We have done this by first identifying the security requirements in the framework, as discussed in the previous chapter. These requirements led us to identifying the threats that an adversary poses on the CRESCENT framework. We have analyzed the possible security loopholes that make the framework vulnerable to many attacks resulting in security violations and possibly, a system breakdown. We later identified the possible threats based on vulnerabilities analyzed beforehand, using threat modeling technique. Threat modeling is a methodology that is used by software developers and researchers to help them find, evaluate and document the threats and corresponding attacks, vulnerabilities the attacks target, and countermeasures that could be taken to patch those vulnerabilities to avoid targeted attacks and ultimately minimize the threats (Bertino et al., 2010). Minimizing the security risks to the engineering solutions is one of the primary goals of threat modeling technique.

3.4.1 Vulnerability Assessment:

A set of indications, *vulnerability categories*, are defined for aiding in identification of threats corresponding to specific vulnerabilities in any system/ framework. These vulnerability categories indicate all the possible vulnerability exploitations that could result due to lack of implementation of corresponding security functions. In Table , we have shown a set of vulnerabilities in our framework and corresponding threat lists, identified for modules within the framework. The set is based on vulnerability categories for web services, defined by Microsoft (Bertino et al., 2010).

The vulnerabilities in the CRESCENT framework as per Table 3.2and the possible threats resulting in case of their exploitation are explained below.

1. **Input Validation:** Web based systems are generally prone to attacks that involve bad or malicious input from the user. Depending on the security level that has been implemented in the system, the users are mostly capable of manipulating system resources, i.e., viewing, creating, deleting or changing, using well-known attacks such as SQL injection, buffer overflow as well as cross-site-scripting. In CRESCENT, the Demand Flow Controller module, receiving all the incoming user requests is prone to such attempts by an attacker.

2. **Authentication:** Users require credentials to authenticate themselves with a system, and access the services. This is possible if they are already registered and their identity is part of the system. This stored identity is cross checked every time the user desires to avail services of

the system. These credentials if stolen by the adversary, could lead to system compromise. This can be done, using network eavesdropping, if the credentials being processed for identification are not encrypted. The modules in CRESCENT do not carry out mutual authentication before working on any request they receive. This could lead to replay attacks, brute force attacks as well as credential theft through network eavesdropping.

3. **Authorization:** Authorization mechanisms are implemented in the system to allow only the authorized users to access system data. Lack of such mechanisms make the data in the system vulnerable to illegal access as well as tampering. The modules in CRESCENT that are responsible for managing data at rest i.e., SLA pools, Workflow Pools, Component Pools must first check whether the requesting module is authorized to access the resource he is requesting for. Then, later he should be issued the resource he requires for performing any function.

4. **Sensitive data:** Data stored in plain form in storages i.e., pools is vulnerable to tampering attacks. The attacker can easily view and retrieve resource of his choice.

5. **Session management:** The sessions for each user is not secure due to lack of encryption mechanisms. This can aid an attacker in eavesdropping and getting desired information regarding the users' requests. The communications between all the modules of the framework need to be made secure.

6. **Parameter manipulation:** The parameters like Query strings, feedback form field inputs, cookie s HTTP header could be manipulated by any attacker, leading attacks like replay attack or even denial of service.

7. **Exception management:** The system should generally be capable of handling exceptions such as attempts leading to denial of service or involuntary information disclosure. CRESCENT does not perform exception management.

8. **Auditing and logging:** In majority systems, the attackers perform malicious activities and cover their track afterwards, due to absence of log protection mechanisms. The system logs should be maintained and

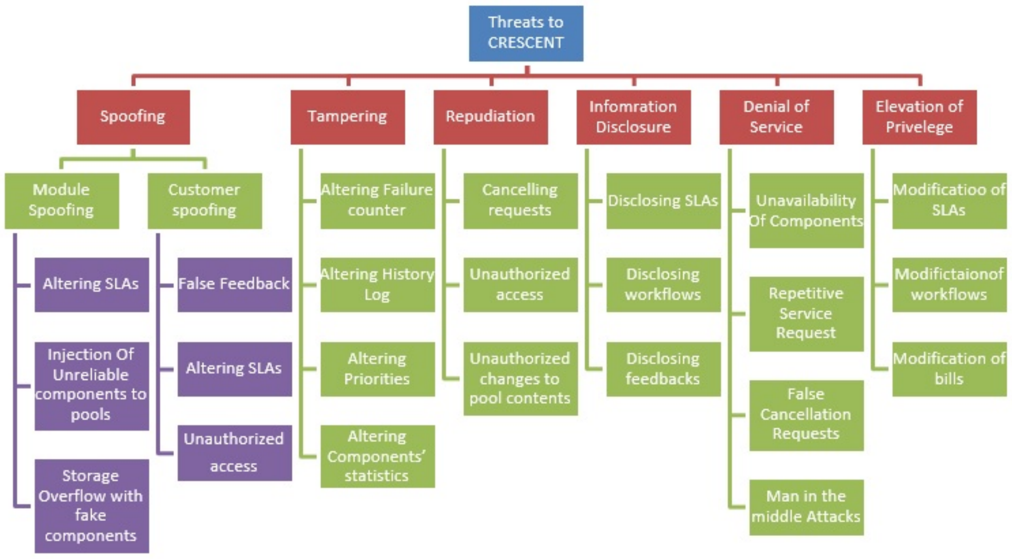


Figure 3.2: Taxonomy tree of possible threats on CRESCENT based on STRIDE

auditing should be done. These audits help prevent non repudiation by customers as well as different modules of the framework.

3.4.2 STRIDE Threat Modeling:

Categorization of possible threats can offer a more focused methodology to conduct the security analysis of the framework. We have analyzed the threats with respect to the objectives and vulnerabilities discussed above, that the attacker can most likely exploit. We have performed the analysis based on Microsoft's description of adversary's objectives using STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege) categories (str,). We will first explain each threat category with respect to our framework with help of a Table 3.3. Based on vulnerabilities mentioned before, the possible threats are as follows.

Now, as the threats categories have been defined, we can form a hierarchy, of possible threats to the framework. Figure 3.2shows the taxonomy of various types of attacks that could occur due to absence of security features through different ways, as shown under each threat category. The diagram is further explained via Attack trees mentioned below:

Attack Trees:

There are also many noted attack graph tools, (mul,) , (Win,) , (tan,), to model attacks. The decision to use attack trees was based on many reasons that include:

1. It uses an easy structured way to describe the actions that an attackers could take to perform a successful attack
2. The model is easy to understand, even for beginners structure
3. The attack trees are presented in a hierarchical, which shows higher-level goals breaking down into sub goals, until a refined form is obtained.

We have used ADTool (Kordy,), for creating attack paths and attack trees for the framework. The ADTool application provides a tool to assess threats from an adversary's perspective. It uses an attack-tree method for assessing how an asset can be attacked by an attacker, what harm he does with his attack, and what measures need to be taken to become immune to that attack (Sathiaseelan, 2013). Each category of threats that has been identified in the previous section, may include further possible threats. Attack trees (Schneier, 1999) help us to identify those threats. ADTool is an open source tool for attack tree generation and modeling. The tool helps create attack trees and perform various types of quantitative analysis of attacks predicted. We have chosen attribute "Probability of success of attack" for our evaluation. The attack trees generated show high probability of occurrence of attacks. While, after application of security mechanisms to cater all vulnerabilities, the probability of success of attacks reduces to zero. We have created attack trees for the threat categories, identified using STRIDE, discussed before in the previous sections. The figure shows probability of each node which is set for the immediate goal-node above to occur and resulting probability shows in the root node. The root probability depicts the overall probability for the occurrence of the attack in ways depicted by the attack tree.

1. **Spoofing:** Figure 3.3 shows, the ways an attacker can spoof user (customer) identity to perform malicious activities such as issuing illegal requests for cancellation of service requests, or spoof a module and perform malicious activities like altering SLAs or billing information. The spoofing could be performed by either stealing user credentials or else performing malicious operations which could be done if attacker adds

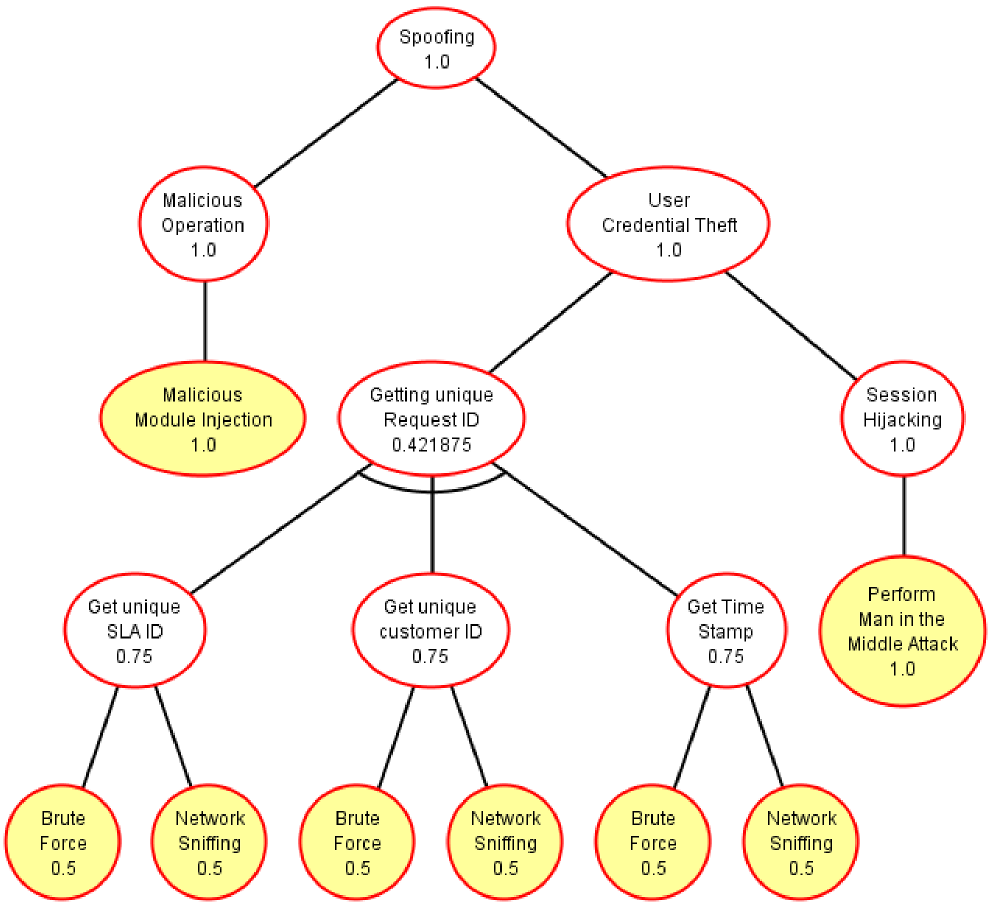


Figure 3.3: Attack tree for spoofing threat in CRESCENT

a malicious module into the framework. Credentials could be stolen by getting unique request IDs assigned to customers by the system. This ID is composed from SLA ID, customer ID and a time stamp. The attacker could get these parameters by either brute force or by network eavesdropping and re-using the eavesdropped parameters for issuing fake requests.

2. **Tampering:** Tampering is the unauthorized modification of data. Modifications in data could be done by tampering the data flow, the SLAs, requests from customers or the data at rest. Data flowing between modules is vulnerable to manipulation by attackers. The adversaries could capture the parameters being exchanged by modules, which are critical for service composition process. These parameters

such as components' statistics provided by Capacity Planner to the Component Manager module and the failure counter being altered and reported after every failed attempt etc. could be modified by the attacker. SLAs could be tampered either by inserting malicious SLAs into the SLA Pool or maliciously modifying the already existing ones. This could be done by modifying the parameters associated with SLAs such as workflow priority or the associated workflow itself. The request coming from user could be altered by an adversary if the receiving Demand Flow Controller module is malicious or under the control of an attacker. This could also be done by modifying the SLA ID, Time stamp or Customer ID. These could further be obtained if an attacker performs a man in the middle attack and retrieves these parameters. The data at rest such as data stored in pools could also be altered by adding malicious components to pools or update the existing components. This can also be done by first inserting a malicious manager module and then modifying the data in pool. Attack tree is shown in Figure 3.4.

3. **Repudiation:** Repudiation refers to the ability of users that they claim to not have done anything that they actually have. Without adequate auditing, repudiation attacks are difficult to prove. An adversary can easily deny performing critical actions in CRESCENT due to absence of audit information regarding all actions being performed. The attacker could perform desired operations and actions after adding malicious module to the system. Absence of digital signatures as well as auditing of operations makes it impossible to track or blame a certain entity for any action. Attacker could also submit false feedback on behalf of customer, send fake cancellation requests or manipulate bills created for a certain service. Attack tree is shown in Figure 3.5.

4. **Information Disclosure:** Information disclosure is the undesired exposure of private data of any application. Such as, if a less privileged user is able to view a file in the system, that he or she is not authorized to, or if he is able to monitor and analyse the data flowing over the network. Information within the system is generally exposed through various ways such as the data is usually added by programmers in hidden fields of forms, which can easily be viewed by any user and manipulated. Comments are also usually added within the web page code, which reveal information about the system and the functionalities it is performing. These comments could lead to revealing of critical information regarding databases, system exception handling

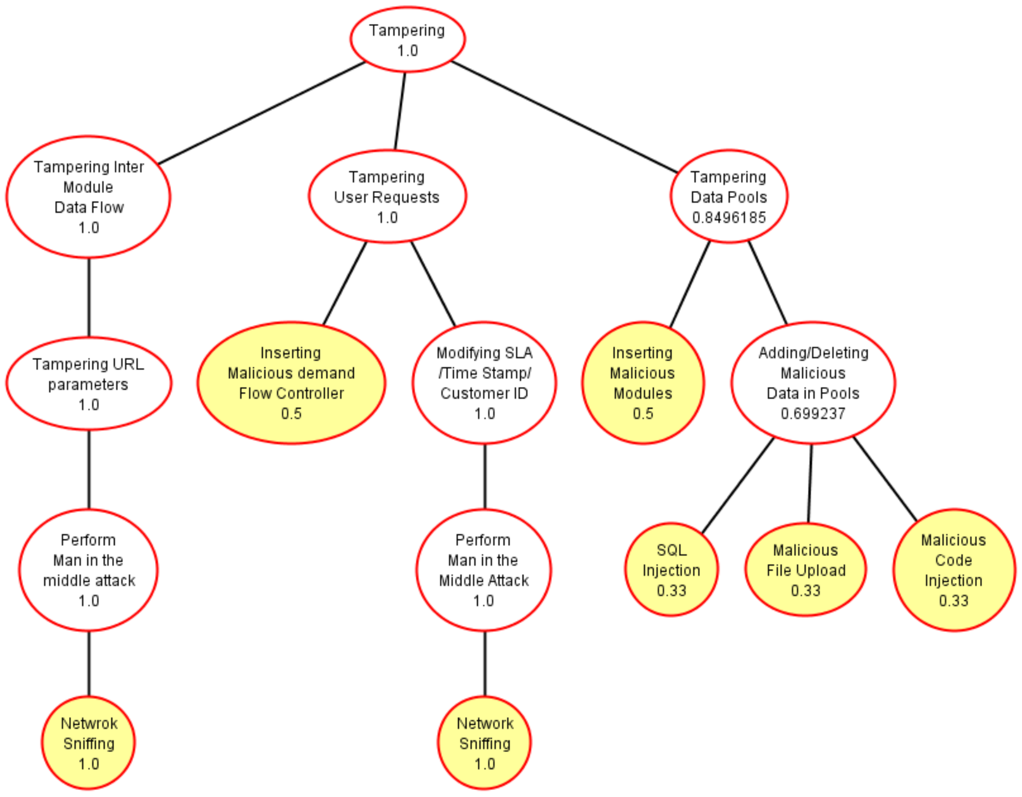

Tampering
1.0
Tampering Inter Module Data Flow
1.0
Tampering User Requests
1.0
Tampering Data Pools
0.8496185
Tampering URL parameters
1.0
Inserting Malicious demand Flow Controller
0.5
Modifying SLA /Time Stamp/ Customer ID
1.0
Inserting Malicious Modules
0.5
Adding/Deleting Malicious Data in Pools
0.699237
Perform Man in the middle attack
1.0
Perform Man in the Middle Attack
1.0
SQL Injection
0.33
Malicious File Upload
0.33
Malicious Code Injection
0.33
Netwrok Sniffing
1.0
Network Sniffing
1.0

Figure 3.4: Attack tree for tampering threat in CRESCENT

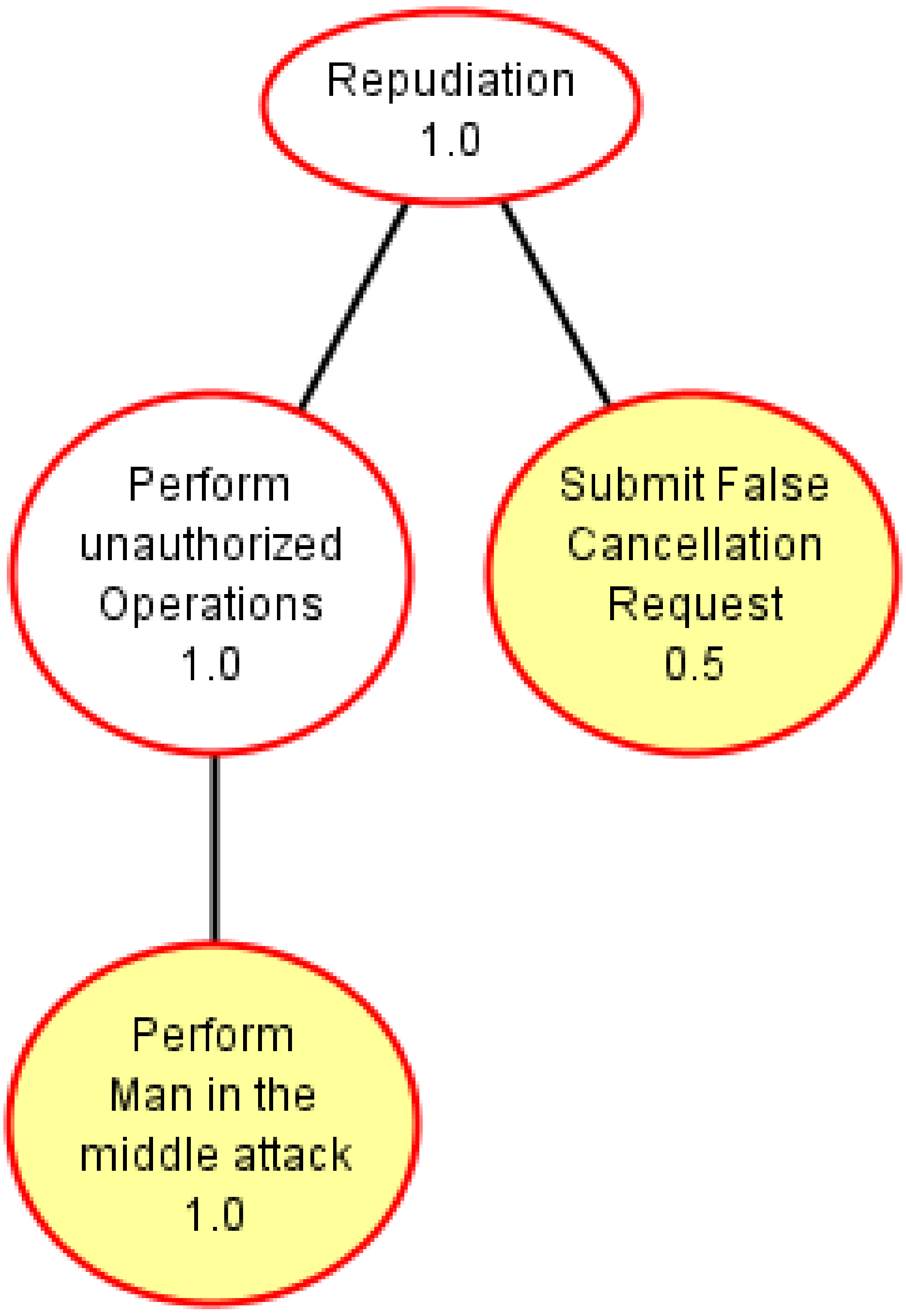

Repudiation
1.0
Perform unauthorized Operations
1.0
Submit False Cancellation Request
0.5
Perform Man in the middle attack
1.0

Figure 3.5: Attack tree for Repudiation threat on CRESCENT

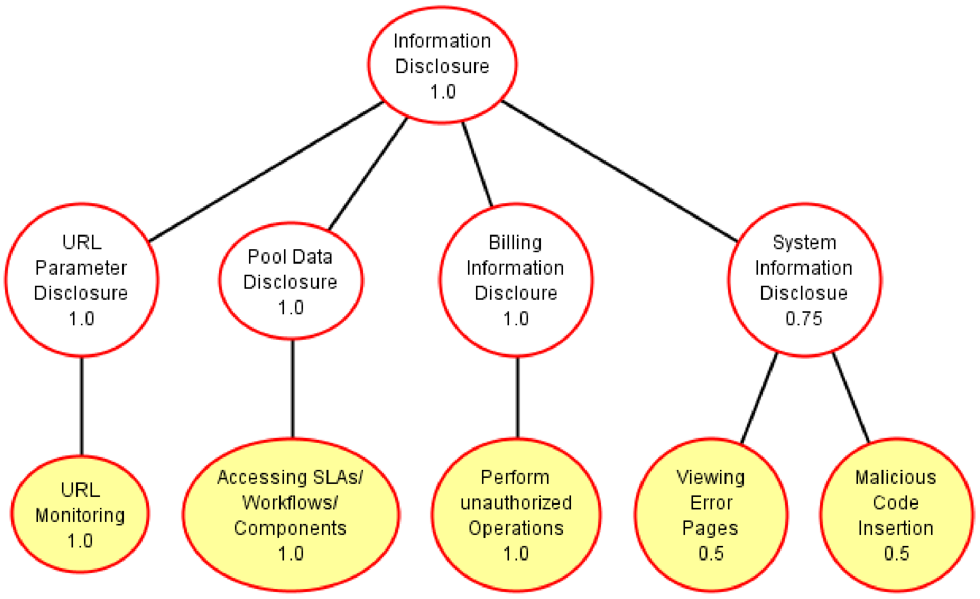

Information Disclosure 1.0
URL Parameter Disclosure 1.0
Pool Data Disclosure 1.0
Billing Information Discloure 1.0
System Information Disclosue 0.75
URL Monitoring 1.0
Accessing SLAs/ Workflows/ Components 1.0
Perform unauthorized Operations 1.0
Viewing Error Pages 0.5
Malicious Code Insertion 0.5

Figure 3.6: Attack Tree for Threat regarding Information disclosure in CRESCENT

mechanisms etc., which could be very useful to the attacker. The information critical for a system must remain undisclosed. In CRESCENT, no such mechanisms are present that could prohibit such involuntary exposure of information. The cookies could be accessed through inspection of URLs as well as monitoring of data flow between web pages through network eavesdropping. Hidden parameters in web pages could be viewed by inspecting the code of web pages by the adversary. Also, user credentials could be stolen from within the dataflow by eavesdropping. Similarly, request parameters could be viewed too, through URLs as proper filtering mechanisms have not been implemented. Attack tree is shown in Figure 3.6

5. **Denial Of Service:** An important feature in web services is constant availability of a service. This factor helps build a reputation of the service provider. Denial of service attacks are crucial for reputation of service provider as well as efficient service provision to customers. Such attacks could be launched on CRESCENT either by flooding the system with requests using bots, that generate automated requests, or compromising the SLA Manager module as well as the Component Discovery Manager module. This module is responsible for finding ap-

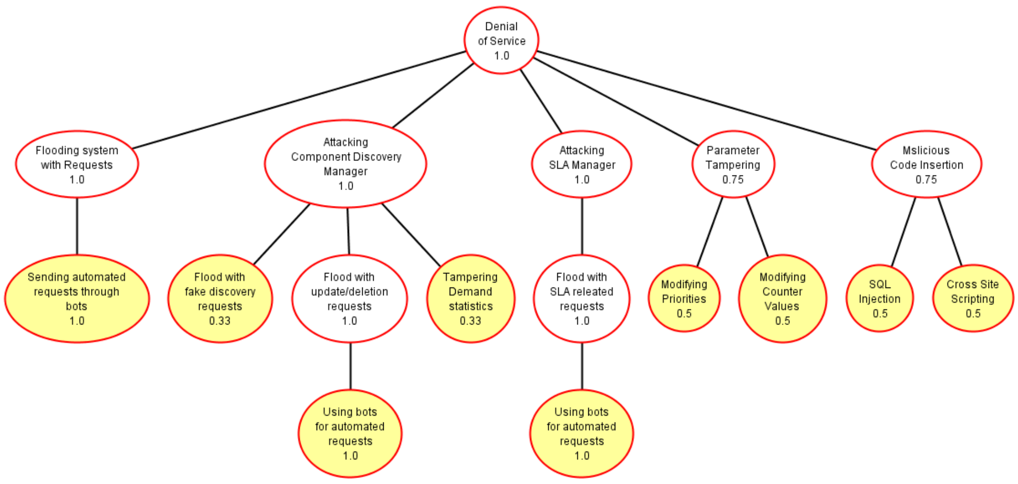

Denial of Service 1.0
Flooding system with Requests 1.0
Attacking Component Discovery Manager 1.0
Attacking SLA Manager 1.0
Parameter Tampering 0.75
Mslicious Code Insertion 0.75
Sending automated requests through bots 1.0
Flood with fake discovery requests 0.33
Flood with update/deletion requests 1.0
Tampering Demand statistics 0.33
Flood with SLA releated requests 1.0
Modifying Priorities 0.5
Modifying Counter Values 0.5
SQL Injection 0.5
Cross Site Scripting 0.5
Using bots for automated requests 1.0
Using bots for automated requests 1.0

Figure 3.7: Attack Tree for Threat of Denial of Service Attack on CRESCENT

propriate components for creation of composition plans for realizing the requested services. The attacker may also insert a malicious Component Discovery Manager module and take control of the discovery mechanism. The request for new components is sent by component manager module. This module could also be spoofed to send large amount of requests to overwhelm the component discovery module. This could again be accomplished by installing bots within the network. Denial of service attack could also be done by tampering the parameters such as request cancellation counter, resubmittance counter or SLA/workflow/component priorities. This tampering may halt the system and eventually disrupt the service. Attacker could also perform SQL injection attacks and cross site scripting attacks to inset malicious data for disrupting the service. Attack tree is shown in Figure 3.7

6. **Elevation of Privileges:** Elevation of privilege relates to a scenario in which a user with less privilege gains the capability of access to a asystem with rights more than its own role's. For example, an attacker with less privileges, might be able to perform actions that require privilege of some modules that are involved in critical functions, and may eventually take control of a highly privileged and trusted process. In CRESCENT, an attacker can take control, or act as a privileged module of the system to perform illegal actions by inserting a malicious module or spoofing the modules already present. Attack tree is shown in Figure 3.8

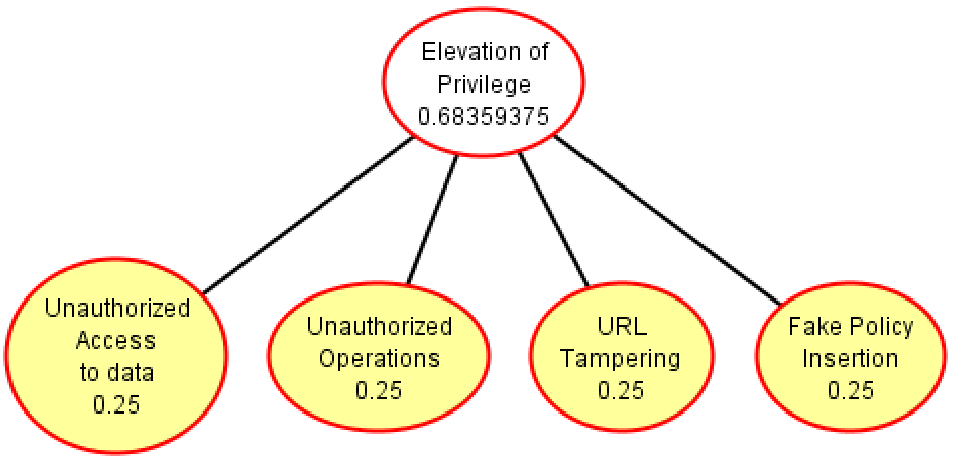

Elevation of Privilege 0.68359375
Unauthorized Access to data 0.25
Unauthorized Operations 0.25
URL Tampering 0.25
Fake Policy Insertion 0.25

Figure 3.8: Attack Tree for Elevation of Privilege Threat to CRESCENT

	CRESCENT	BFT-WS	PERPETUAL	BASE	THEMA	(Zhao, 2007a)	FACTS	(Onditi et al., 2008)
Composite Web Services Delivery	√					√	√	√
Service Logic/Execution Separation	√							
Adaptive Workflow Construction	√						√	
Automatic Failed Requests Resubmission	√						√	
Components BFT via Replication		√	√	√	√			
Components BFT via Parallel Provisioning	√							
Dispatchers BFT via Replication	√		√			√		
Asynchronous Invocation	√		√					
Asynchronous Processing	√		√	√	√	√		√
Fault Isolation	√		√				√	√
Components Error Recovery	√						√	
Dispatchers Error Recovery	√							
Automated Capacity Management	√							
Automated Components Discovery	√							
Automated Cancellation Management	√							
Client Isolation	√			√	√			√

Table 3.1: BFT Web Services Delivery Approaches Comparison

Table 3.2: Vulnerability Categories (Defined for Web Services by Microsoft) and Associated Threats in CRESCENT

Vulnerability Category	**Attacks**	**Vulnerable Modules**
Input validation	Buffer overflow; cross-site scripting; SQL injection	Demand Flow Controller
Authentication	Network eavesdropping; brute force attacks; Dictionary attacks; credential theft; cookie replay;	All modules
Authorization	Disclosure of confidential data; data tampering	Work Flow Manager,Demand Flow controller, Pool Managers
Sensitive data	Accessing sensitive data in storage pools; network eavesdropping; data tampering	Workflow Pool, Component Pool, SLA Pool
Session management	Session hijacking; session replay; man in the middle	All Modules
Parameter manipulation	Query string manipulation; feedback form field manipulation; cookie manipulation; HTTP header manipulation	Component Monitor, Feedback Module, Component Manager, Workflow Manager, SLA monitor,Billing Module, Cancellation Module.
Exception management	Information disclosure; denial of service	Demand Flow Controller, Capacity Planner
Auditing and logging	User denies performing an operation; attacker exploits an application without trace; attacker covers his or her tracks	Adaptive Composer, Dispatcher, Output Controller, Managers Modules.

Table 3.3: Threat categories with definitions, related to CRESCENT and Associated Vulnerability Categories

Threat Category	**Definition**	**Vulnerability Categories**
Spoofing	The attacker might get an illegal access to the system or any specific module and misuse or disrupt it by spoofing identity or authentication information of the customers as well as the modules. The modules performing critical functions such as adaptive composer, which is involved in executing the composition plans, if spoofed can cause the worst damage.	Input validation, Authentication, Session Management
Tampering (Integrity threats)	This refers to the unauthorized modification of the data i.e. data being exchanged (users' authentication data, users'request specific data i.e. SLA ID etc.), data being stored in pools (SLAs, components etc.) or data being processed or generated at various points within the framework (bills, feedback etc.)	Authorization
Repudiation	The user may deny performing certain actions or operations e.g. an attacker or a malicious user may deny or cancel a legitimate request from a customer.	Parameter Manipulation, Auditing and logging
Information disclosure	The attacker may expose the critical confidential data such as customers' SLA ID, unique ID and other unique information required for processing the customer's request.	Authorization, Parameter Manipulation, Sensitive data, Session Management.
Denial of Service	The attacker may overwhelm the system with repetitive similar requests thus resulting in denial of service due to unavailability of required components for composition. This could result in service unavailability.	Session management, Authorization
Elevation Of Privilege	The attacker might spoof the identity of customers or the service provider to gain control of the modules by issuing illegitimate requests. The requests may cause the cancellation of legitimate requests. The attacker might also delete or modify the components within the pools using elevated privilege of the manager modules of the framework.	Parameter manipulation, Exception Management

Chapter 4

Design and Evaluation of Proposed Solution

Chapter 5 includes complete description of the solution that has been proposed, CRESCENT+. The security issues analyzed in the previous chapter through threat modeling and their visualization been done through attack trees, were catered for using various security modules. Each security module handles a different security feature employing a specific security mechanism. Execution flow of each security module with respect to CRESCENT has been described. Later, we have also added the whole work flow of the CRESCENT+ framework for service delivery. After the description of workflow, we have evaluated the addition of security modules with respect to the previously mentioned threats and vulnerabilities of the framework. The functionalities offered by security modules effectively cater to the identified security requirements of the framework.

4.1 CRESCENT+ Proposed architecture

We have proposed CRESCENT+ framework that includes security modules for incorporating required security features into the framework. This is necessary for ensuring secure and reliable dynamic web service composition as per customers' requests. The proposed solution mitigates all the threats faced by CRESCENT that have been identified in the previous chapters. The threat modeling performed on CRESCENT framework helped analyze the criticality of incorporating security features into the management framework, for reliable and durable service provision. The architecture of CRESCENT+ is shown below in Figure 4.1.

4.2 Security Modules with Execution Flows:

The security modules to be added offer all the mainly required security features which are strong authentication, fine grained authorization, data encryption and data integrity. For providing these features,we have used some well known public standards namely SAML (SAM, 2005) , XACML (Rissanen, 2010) and IETF RFC (T. Hayashi, 2017). The security modules added for this purpose are described below with the process of their execution:

4.2.1 Authentication Module (AM):

The purpose of this module is to ensure and verify that every entity interacting with the framework or is a part of the framework, has a valid identity. The entities could be the customers who may request for a service or the inter framework modules. The authentication module is further divided into two sub modules that together facilitate the authentication process, which are, *Strong Authentication Module* and Certificate Authority.

1. ***Strong Authentication Module:*** This module performs authentication utilizing IETF mutual authentication protocol. It also provides additional functionalities of certificate verification using Certificate Authority (CA) and identity verification using Identity Management Module (IDM). Moreover, it issues and stores SAML authentication tickets for the users and the framework modules.

2. ***Certificate Authority:*** CA provides the facility of digital signatures and issues certificates based on XML Key Management Specification (XKMS). The certificates contain the public key and the credential information necessary for verification of certificate holding entity for purpose of authentication. CA is responsible for verifying the authenticity and managing certificates as well as the keys of entities. Integration of CA with XKMS provides an easy and simple XML based protocol to manage key/certificate information. Therefore, making it understanding of underlying complex knowledge of CA and its syntax, a redundant activity.

Workflow:

The Figure 4.2 below show the workflow of authentication phase of customer with the Demand Flow controller (module associated with receiving requests from customers) of CRESCENT+ framework, through Authentication Module. Same procedure will be followed for authentication of intra-framework entities.

1. DFC receive login request from the incoming user.
2. It forwards it to SA for checking its validity through IDMS.
3. IDMS sends back a response if the user is proved to be valid.
4. The user submits is certificate to SA for verification through certificate chain validation.
5. Certificate validation request and response messages are sent to and received from CA.
6. If both verifications are successful, SA issues a SAML ticket together with a SAML authentication response which contains attributes that are a proof of user authentication and attributes of user and the issuer.
7. This SAML ticket is stored by user locally as well as by SA for future verification.

This module will help with mutual authentication of customer with the framework as well as CRESCENT modules among themselves before interacting and exchanging critical data among them for proceeding with carrying out of certain user request. This will eliminate the risk of spoofing of modules. It will ensure if a certain module is authorized to make a certain request. Therefore, modules will be first authenticated with the system and among them, and then later allowed to perform operations on user request for certain service.

4.2.2 Access control Module (ACM):

Access Control Module is responsible for checking the authorization of the requesting entity, whether it is authorized to request access of a certain resource, thus it ensures that data is protected from unauthorized access. It is further composed of three sub modules, namely, Policy Administration Point, Policy Enforcement Point and Policy Decision Point.

1. ***Policy Administration Point (PAP):*** PAP is the part of access control module that helps users to define and create policies for providing access to users based on their role. The administrators can create policies, update them and delete them at any time as per their system requirements. The policies that are created eventually are stored

locally within the system and checked for user's access as soon as the access control module is contacted.

2. ***Policy Enforcement Point:*** PEP is generally responsible for acting as an intermediary point between other two entities, namely Policy Decision Point and the Policy Administration Point. The PEP collects the users request to access a resource and converts it into a SAML-Wrapped XACML authorization decision query. It then sends it to the PDP. After a decision is made against a request, that authorization decision is received by PEP from PDP and converts it into the understandable format for the system.

3. ***Policy Decision Point:*** PDP makes an authorization decision based on the previously defined policies by PAP. The request is received and corresponding policies from repository are retrieved. The policies are checked for an ALLOW or DENY response. If the permission is granted, the PDP requests the Key Distribution Module to fetch the requested resource using a SAML Attribute. If the decision is disallow, the PDP forwards the response to PEP, which is then forwarded to the requesting entity.

Whenever a request will be sent to a Manager module i.e., either SLA, component or workflow, the Access Control Module will ensure, that the request is valid and whether the module is authorized to have access to the resource. If the Access Control Module issues an Allow response, then the access to the requested resource is granted, or if its a Deny, the request is denied. This module will help prevent illegal or unauthorized usage of framework resources.

Workflow:

The description below together with Figure 4.3 presents the workflow of authorization phase between SLA manager and the SLA pool, through Access Control Module. Authorization is going to be verified whenever SLA Manager attempts to access SLA from SLA pool or update/delete/modify it. Similar procedure will be followed when Workflow Manager and Components Manager modules access their corresponding pools.

1. SLA Manager makes an SLA access request along with the SAML authentication token issued by Authentication Module. Request is received by PEP.

2. PEP checks the validity of SAML token through SA of Authentication Module.
3. If the token is valid, the PEP then issues an XACML-SAML based authorization decision query to PDP. The request contains SLA ID and action required and other XACML attributes.
4. PDP retrieves XACML policy from the policy repository and based on attributes of authorization query, decides if the requesting entity(SLA Manager) is allowed the access to specified resource or not.
5. PDP then forwards the decision to PEP for enforcement.
6. If the decision is DENY, it is forwarded to PEP. If it is ALLOW, a SAML authorization response is generated and sent to requesting entity (SLA Manager). Also, an attribute query request is sent to SLA Pool, where the SLA to be accessed resides.
7. As the SLA pool is encrypted, the attribute query request is then forwarded to KDS instead.

4.2.3 Encryption Module (EM):

The modules of the CRESCENT framework interact with encryption module for encrypting and decrypting of resources stored in pool/storage units. This module is involved in creating and managing AES-symmetric keys for encryption of data in SLA pool, workflow pool and component pool thus ensuring integrity of stored SLAs, components and workflows. The encryption service together with key management is managed by two submodules namely *Encryption/Decryption Module (EDM)* and the *Key distribution Module (KDM)*.

1. ***Key Distribution Module: :*** KDM manages keys for all storage units/pools of the framework. AES key is generated separately for every pool and mapped to the authorized module accordingly in KDM database. Whenever a module is granted permission to access data of a pool, the PDP issues an attribute query request to the KDM for the retrieval of specific encryption key of that from the KDM database. KDM verifies and validates the issued request using IETF protocol through CA and looks for the keys present in KDM database against the attributes stated in the request. The attributes may include the

requesting entitys identity information and the resource ID to be accessed. The retrieved key of the pool resource is passed to the encryption service for further operation which might be encryption/decryption (Encryption in case the data is to be stored, decryption if the data is to be retrieved).

2. ***Encryption/Decryption Module:*** As soon as the KDM provides the key for encryption or decryption, the encryption module encrypts/decrypts the resource accordingly. It handles the operation based on incoming request from PDP.

Workflow:

The Figure 4.4 shows the workflow of encryption/decryption phase between SLA manager and the SLA pool, through Encryption Module. After the authorization check is made by Access Control Module, the SAML attribute query request issued by PDP is fetched by KDS for retrieving the key from Key DB and decrypting the SLA needed by the SLA Manager from SLA Pool.

1. The KD module receives the SAML attribute query request from Access Control Modules PDP.

2. KD module verifies with Authentication Module (CA), if the request is actually made by PDP (to ensure non-repudiation). CA retrieves issuers signature from attributes of request query and verifies the owner of the request.

3. If verification of attribute query request, KD gets the key from Key DB.

4. The Key of SLA Pool is retrieved and sent to Encryption/Decryption module.

5. E/D module uses the key and decrypts required SLA record from SLA pool.

6. The decrypted SLA is then sent to SLA manager enveloped in SAML attribute response.

4.2.4 Identity Management System (IDMS)):

IDMS is responsible for verifying identities of the customers and modules of CRESENT framework. It manages identities of all the involved modules and the incoming customers. The modules are initially registered with the IDMS and credentials are stored. These credentials are later checked verified whenever the module is requested to exchange data for accomplishing the request of a user. After verification, it issues a SAML authentication assertion to prove the authenticity of modules, which is used by SA to further verify users certificate through CA.

Figure 4.5 gives a holistic overview of modules interacting with each other for providing necessary security features to the CRESCENT framework.

4.3 Workflow of CRESCENT+:

The workflow of the architecture proposed is explained below. First, we will identify some pre requisites for secure composite service delivery.

4.3.1 Security Pre-Requisites:

1. All the modules of CRESCENT+ go through registration process, prior to receiving and realizing any composite service request from customers. The modules register themselves with the IDMS of the framework, which stores and manages their credentials.

2. The SLA Manager stores the created SLAs in the SLA Pool, after encryption with a data encryption algorithm.

3. The Workflow Manager stores the created and updated workflows in Workflow Pool in encrypted form.

4. Prior to forwarding service request to the next module for operation, every module is authenticated by the IDMS

5. Every two modules mutually authenticate themselves with each other, before exchanging service related information.

6. All the communications being carried out are using AES session keys.

4.3.2 Workflow of Framework in response to a Service Request:

1. The customer requesting for a service, logs in to the system. The user is authenticated based on his provided credentials and those stored on the IDMS.

2. The customer then proceeds with SLA creation. He send his credentials to the SLA Manager (SM) for creating SLAs. The customer is first authenticated with IDMS and then the required SLA is created [Ref: Figure 4.6]

3. For carrying out the management of workflows in the Workflow Pool, the Workflow Manager is first authenticated by the IDMS. It can later create, update or delete the workflows. The stored workflows are also encrypted with some data encryption algorithm. The Workflow Manager (WFM) then issues a component request to Component Pool Manager (CPM). The request is approved by Access Control Module.

4. SLA Manager forwards the request to Demand Flow Controller (DFC) which is first authorized by the Access Control Module (ACM). The DFC, processes and prioritizes the incoming requests based on their customer's SLAs which it retrieves from SM. These prioritized requests are then forwarded to Adaptive composer (AC) Module. DFC also maintains a failure count [Ref: Figure 4.7].

5. The AC module processes the requests in a FIFO manner. It generates the suitable composition plans for each request and submits to Dispatcher. AC retrieves components from Components Pool (CP) by requesting CPM.CPM forwards the required components to AC after authenticating the request. AC then generates the composition plan, which is then submitted to the Dispatcher for realization [Ref: Figure 4.7].

6. If CPM does not find the components required by AC in CP, it then first mutually authenticates the Component Discovery Manager (CDM) and requests for discovery of new components. The CDM requests the Service Registries, which respond with new components. These are then forwarded to AC.

7. The CDM is also involved in workflow variation process. If it finds different components than required, it also presents workflow variations to the WFM. The WFM stores these workflows in the WP.

8. Component Monitor (CM) periodically performs the capacity evaluation of the components in the CP and sends the updated statistics to the CPM [Ref: Figure 4.8].

9. SLA Monitor and Component Monitor submits the monitoring results to Dispatcher. Dispatcher then executes the composition plan received from AC and submits the output to the Output Controller (OC) [Ref: Figure 4.7].

10. In case the service request needs to be cancelled, the Dispatcher sends a cancelation request to the Cancellation Manager, which starts the process of cancellation and forward the request to Billing Manager. The Billing Manager generates the bill based on the utilization of the service and responds with updated info to the Dispatcher [Ref: Figure 4.9].

11. The OC prioritizes the service responses, and sends the execution results and billing information to the customer.

The CRESCENT+ in addition to enhancing reliability of composite web services, also ensures their security in terms of confidentiality, integrity and authorized access. Composite services are provided on the fly to meet the demands of the requesting customers.

4.4 Experimentation and Evaluation:

We identified security requirements of CRESCENT in Chapter 3 of this document. Later, in the same chapter we discussed the possible threats to the framework using STRIDE methodology. In our proposed secure framework, CRESCENT+, we have addressed the security requirements and threats against different system entities of the framework, using different security mechanisms. The incorporated security modules fulfil security requirements of individual system entities as well as the whole web service composition process, with-respect-to confidentiality, integrity, repudiation and availability, thus protecting these entities from threats, such as masquerading, eavesdropping, unauthorized access, and denial of service. The proof of concept taht the threats have been catered is shown in teh section below.

4.4.1 Evaluation based on STRIDE:

As previously explained, we have added a number of security modules that provide important security structure to the CRESCENT framework. In this section, we have explained how these modules actually cater the threats to the various modules of the framework, as depicted by attack trees. The security mechanism used and the vulnerabilities they cater for, are shown in Table 4.1.

Table 4.2 shows the types of attacks that are mitigated by introducing the above described security modules into CRESCENT framework.

4.4.2 Attack Trees Post-Security Module Addition:

We had previously generated a number of attack trees corresponding to the threat categories identified using threats. The attack trees showed the possible ways an adversary is likely to use, to perform unauthorized actions and access critical data or modify the data, affecting eventually the performance and user dependability on the composite service delivery provided by CRESCENT. Below, we have discussed the security mechanisms employed corresponding to every attack tree previously generated to explain the attack scenarios. The probabilities of attacks are shown to have been reduced after addition of countermeasures to the framework in the form of security modules. Probability "1" for each countermeasure indicates, the strength of mechanism used, using various industry standards, that cant be bypassed.

Spoofing:

As per the attack tress in figure 4.10, the ways to perform the spoofing attack on the framework included the following scenarios:

- **Network Eavesdropping** and **brute forcing** to get Customer ID, SLA ID and Time stamp of a specific customer request, to get the unique customer ID or SLA ID for future unauthorized modification or unauthorized access to framework. This vulnerability has been catered through usage of secure Session IDs for each user over https, to transmit the data in encrypted format having time stamps of limited life time. Advanced security mechanisms including captchas to discourage brute force have been incorporated. Also, input validation has been applied.

- **Man in the middle attacks** are catered using HTTPS over the internet and proper authentication mechanism through CA. The entities interacting with the system are properly authenticated, SAML authentication tokens are issued and their certificates are verified through trusted certificate authority.

- An attacker might interact with the resources with a faade of being an authenticated manager module, authorized to access that particular resource. For that purpose, access control module ensures that the module is firstly authenticated via CA by contacting the Authentication Module, and later checks its permissions policies described by PAP. Only then the access to a resource is granted. Therefore, the possibility of a replay attack is completely reduced to zero.

Tampering:

As per the attack tress in figure 4.11, the ways to perform the tampering attack on the framework includes the following scenarios:

- Malicious unauthorized entity could pretend to be a demand flow controller and receive requests from customers and tamper the data or save credential information to perform spoofing. This is catered by using mutual authentication using CA as well as session IDs and tokens for each user, ensuring a secure session. Unauthorized manager modules, may issue requests for accessing data in storage pools and tamper it for effecting performance. Similar mutual authentication is applied as well as access control to discourage unauthorized access to any critical data pertaining to service delivery.

- The modules will only be allowed to modify the data or add a new data element to pools, if corresponding policies as per the allowed permissions exist in PAP.

- Counters for all failed and resubmitted requests are maintained by the framework to access performance time to time and apply counter measures to cater to the issues. The counters if modified could result in decreased performance, eventually causing denial of service to customers. Also, priories associated with SLAs as per their demand and usage could also be altered. For this purpose, the maintained data of pools and parameters associated with service requests is secured in encrypted form using AES keys to prevent any tampering.

Repudiation:

As per the attack tress in figure 4.12, the ways to perform the repudiation attack on the framework includes the following scenarios:

- Adding malicious modules to performed unauthorized activity and removing the activity tracks from logs could result in repudiation from performing specific activity. This is catered through mutual authentication and using SAML assertions to prove the issuer of any request regarding service delivery process. Also, the auditing and logging mechanism is secured using access control mechanism.

- Prevention of submitting false feedbacks and cancellation requests to repudiate is ensured by providing process access control. The manipulation of bills for purpose of repudiation is prevented by SAML assertions as well as appropriate auditing and logging of every activity pertaining to a user or a module of the framework.

Information Disclosure:

As per the attack tress in figure 4.13, the ways the framework is likely to disclosure important information related to service delivery activities includes the following scenarios:

- Network eavesdropping to get credential information or other unique information pertaining to users is prevented using https protocol to ensure secure session over the internet and the data travels in encrypted form.
- Every form of data disclosure through unauthorized access is prevented using access control module that allows access based on policies corresponding to every user and entity of the framework.
- Information disclosed on error pages regarding system, specific methods and architecture is properly controlled through secure coding.

Denial of service:

As per the attack tress in figure 4.14, the ways the framework is likely to deny a service to a user includes the following scenarios:

- The access to every module is restricted through access control module thereby preventing any outsider to issue requests on behalf of legitimate module.
- The logs of requests are maintained together the Ip address of the customer. A threshold is maintained for every ip address to be able to request a certain number of requests at a time. The requests are later moved to least priority by the framework.

- Proper input validation will ensure prevention of any malicious script insertion through form entry points and URLs. Therefore automated malicious activities are restricted.

Elevation of Privilege:

As per the attack tress in figure 4.15, the ways the adversary is likely to access resources or modify module functionality through escalation of privilege includes the following scenarios:

- Every request issued for resource access is properly authorized by Access control module after authentication of issuer through CA of authentication module. Therefore, restricting access to unauthorized requester.
- Authentication module ensures proper authentication of every module based on IDMS information and certificate verification through CA. And every request made is checked against its corresponding allowed permissions.

Following Table 4.2 shows the summary of attacks mitgated through addition of security modules in the framework.

4.4.3 Performance and Reliability Evaluation:

The work in (Elgedawy, 2014) showed that CRESCENT outperformed existing approaches for composite web service delivery. Hence, in our research work, we will focus on the comparison between CRESCENT and CRESCENT+ to show the effect on the reliability and throughput. We also include in the comparison, the basic industrial approach (denoted as NO-BFT/NO-Security) doesnt ensure BFT either for composite web services or modules involved. It also does not handle security issues. So such approach resembles submitting BPEL scripts to BPEL execution engines.

Performance Model:
For sake of comparison of above mentioned three approaches, we have selected a performance model consisting of two queuing system, he dispatcher queueing system and the components queuing system.

- The dispatcher queueing system is responsible for handling the incoming requests from the customers with an input rate of (λ) and as a result processes requests with a rate of (μ_d).

- The component queueing system is responsible for handling requests that are coming from the dispatcher queueing system and processes them at a rate of (μ_c).

Any modelling system tool could be used to create simulations with our provided values. We have used *Matlab SimEvents simulator*. This simulator provides us with average response time for each of the techniques and from that response time, we calculate the throughput as inverse of sums of averages. We have performed experiments for varying values of inputs and plotted them. As the values are random, the results are therefore not exact but the trend that they create should be used and judged instead for a better idea. We have used values for a simple web service generated from three basic tasks. Its demand was set to be λ=30 requests per minute. To calculate μ_d and μ_c for each approach, service time is first calculated and then the service rate, (inverse of service time). Service time is calculated as sum of time taken for processing and network overhead. Network latency is the major factor that creates a limitation in networks and resultantly effects response times of services (Mao et al., 2008) (Amir et al., 2007). After computation, service rates are put into simulation tool along with the input rate and other required attributes. Computations for CRESCENT+ and CRESCENT will be similar, however an extra security overhead is added, which is computed as the extra number of messages exchanged due to proposed security protocols multiplied by the network latency. At beginning of communication, CRESCENT+ requires at most 10 message exchanges per module to satisfy the security requirements. However, after steady state, it only needs 2 message exchanges. Hence, we computed the overhead in the steady state case. Simulations are run for a time period of 24 hours.

Reliability Comparison Experiments:

We have assumed the probability of failure of a given component/module is γ, the corresponding success probability is 1- γ. Such failure probability could be resulting from any sources; it could result from physical failure, computational errors, SLA violation, or security. We run the experiments using n=4 (n being the number of replicas) for both CRESCENT and CRESCENT+ approaches, as the work in (Elgedawy, 2014) indicated this is a practical component cluster size. From expected failure probability, success rates are calculated; with failure probability resulting from both components and framework modules. However, to compute the failure probability in CRESCENT+, we adopted the security quantification analysis proposed in (Madan et al., 2004). Using those quantified probabilities and the CRES-

CENT failure probabilities; we managed to compute CRESCENT+ probability failure. That is the probability a system is still failing provided that the security threat is stopped. Standard conditional probability computation is adopted. The Figure 4.16 shows the results obtained.

Figure 4.16 Reliability Comparison shows that CRESCENT+ has increased delivery success rate when adopted as it ensures the BFT for the modules as well as the framework components. It also reduced the failures resulting from security threats.

Performance Comparison Experiments:

To compare between the through puts of the three approaches, first we run the simulation when having $\lambda = 30$ requests per minute as indicated before. This is to resemble low demand case over the composite web service. Results are shown in Figure 4.17.

Figure 4.17 shows that that an increase of number of components has no effect on the NO-BFT approach, and that the NO-BFT approach has high throughput as there is no overhead for replicas synchronization or security protocols. However, the CRESCENT approach managed to increase the number of components to improve the performance. We can see, as the number of components increases, throughput is reaching a high steady state. However, the CRESCENT+ approach shows a similar behavior to CRESCENT but with fewer through puts due to the overhead of the security protocols. However, we argue that CRESCENT+ still provides acceptable performance, especially in the peak areas. To such effect, we repeated the experiments when $\lambda = 100$ (Figure 4.18) and $\lambda = 300$ (Figure 4.19). As we can see in the following figure, the NO-BFT approach could not handle the high demand, but both CRESCENT and CRESCENT+ managed to increase the throughput using the components parallel provisioning. However, CRESCENT+ will always provide fewer through puts when compared to CRESCENT due to the security overhead.

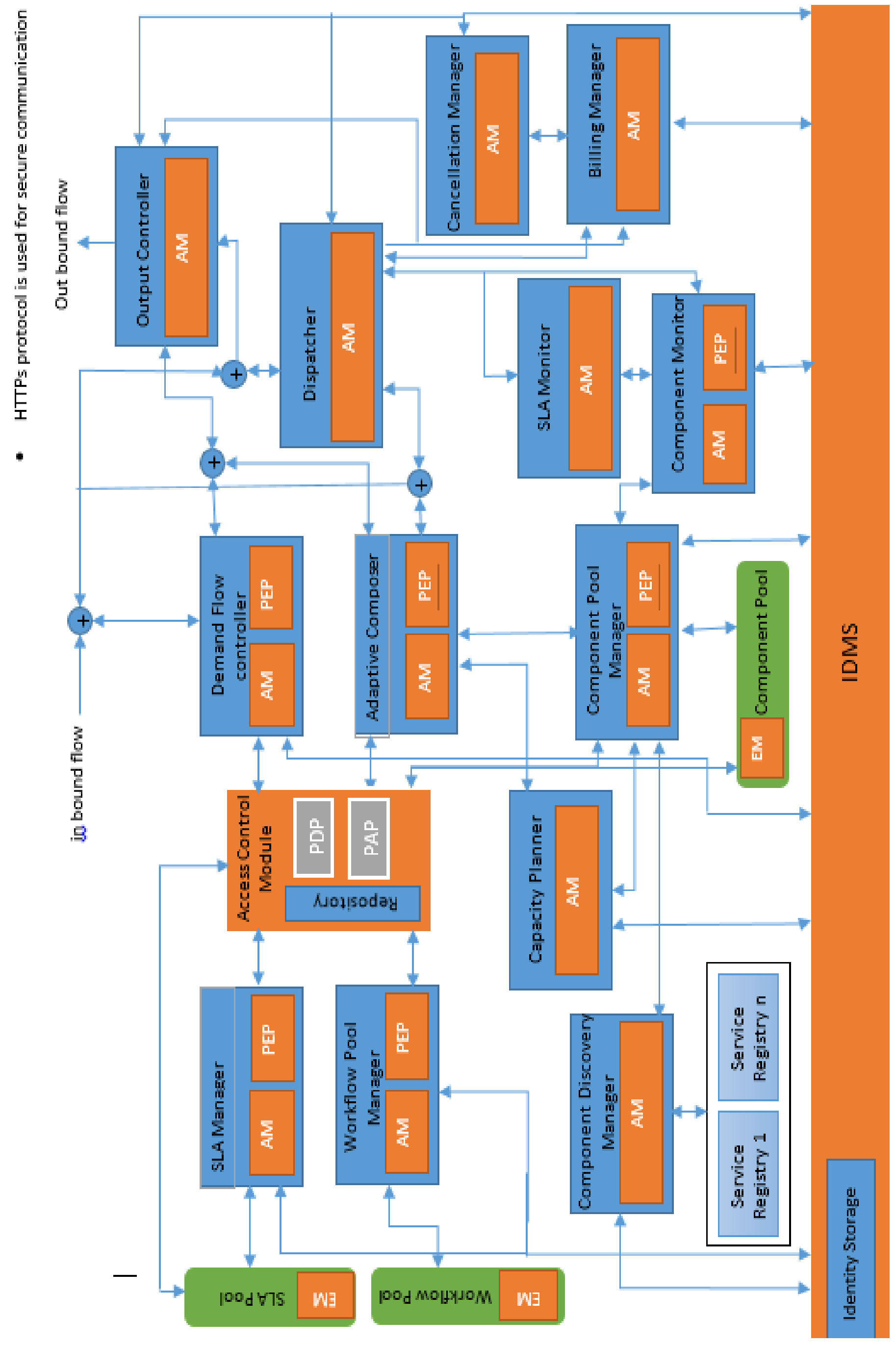


Figure 4.1: Proposed Architecture for CRESCENT+

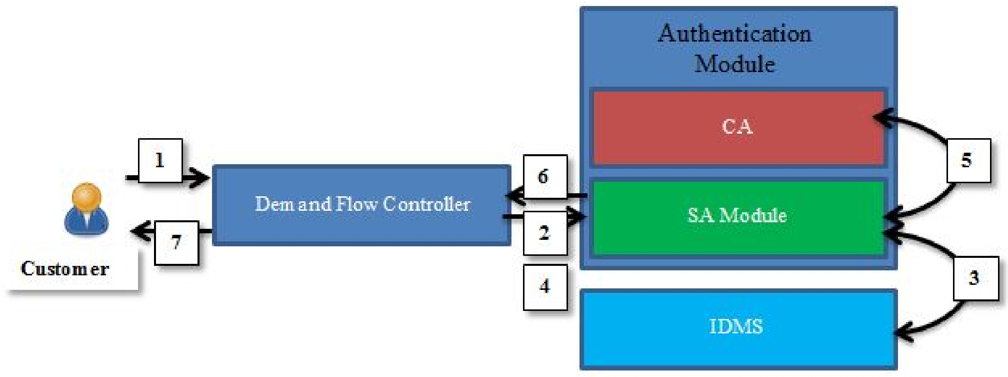


Figure 4.2: Workflow of Authentication Process

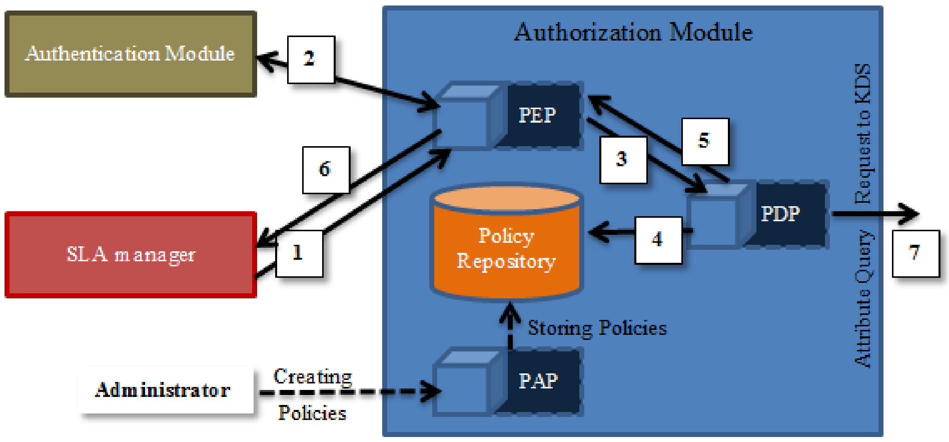


Figure 4.3: Workflow of Authorization Process

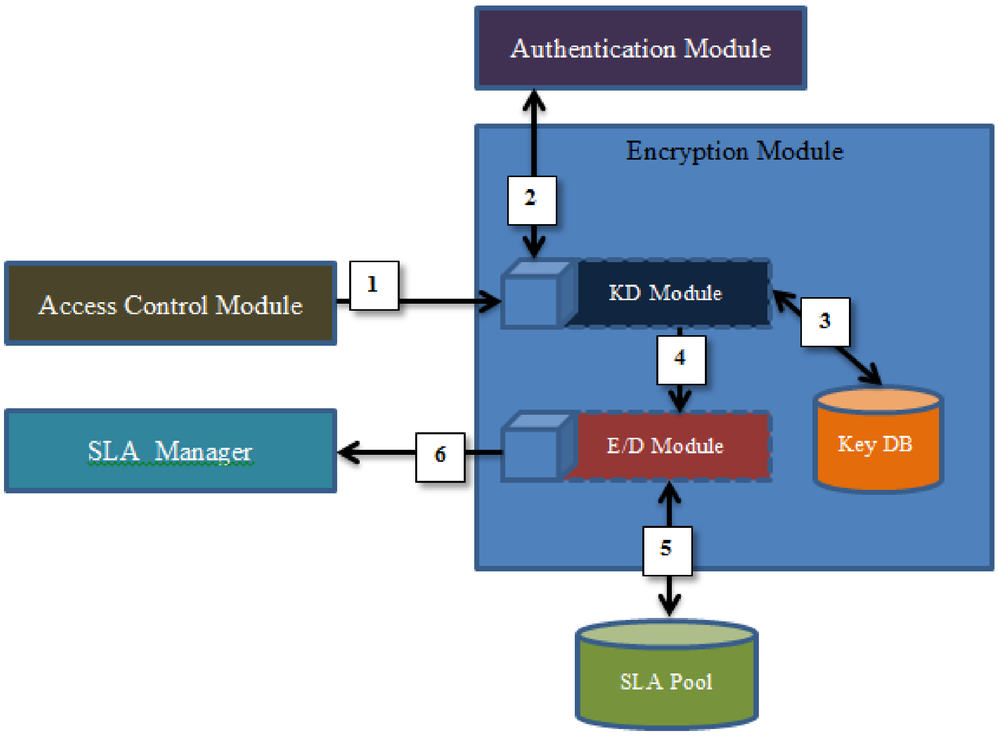


Figure 4.4: Workflow of Encryption/Decryption Process

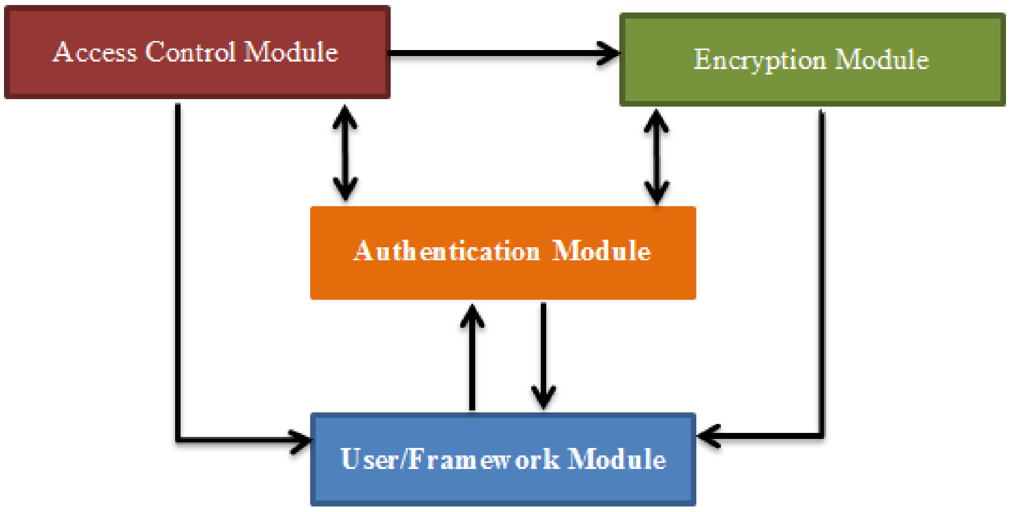


Figure 4.5: Overview of Security Structure of CRESCENT+

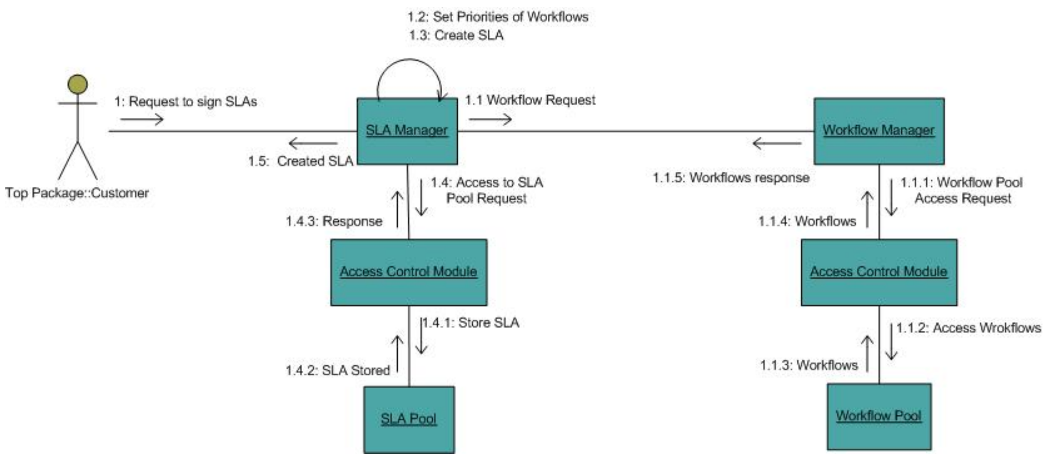


Figure 4.6: Collaboration Diagram for SLA Creation Process

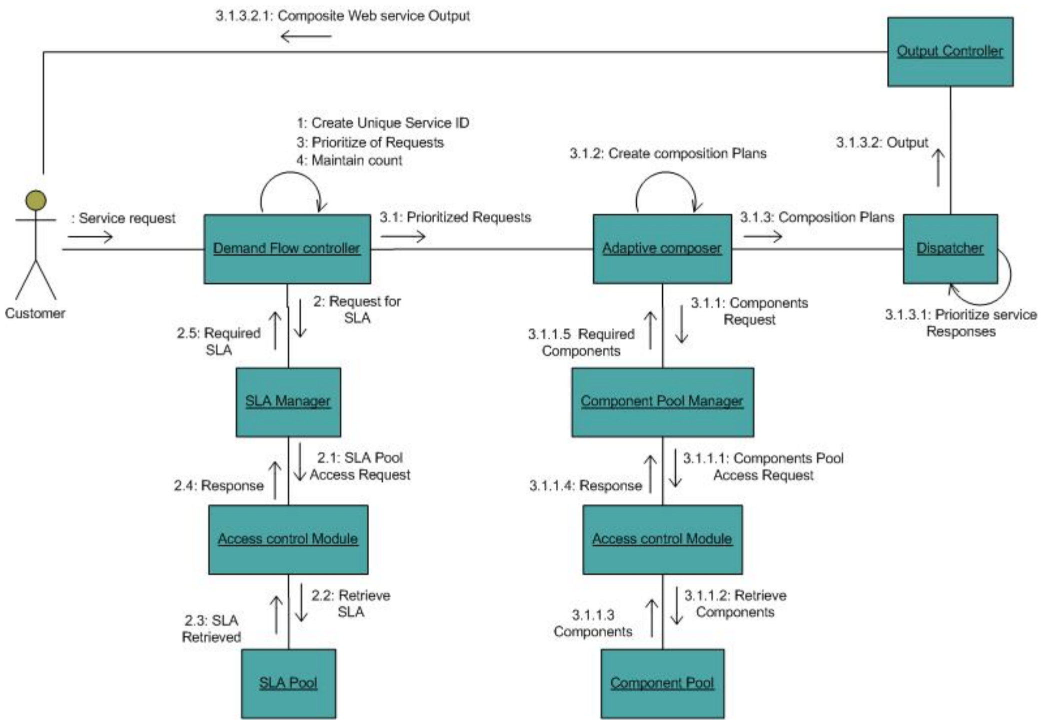


Figure 4.7: Collaboration Diagram for Service delivery

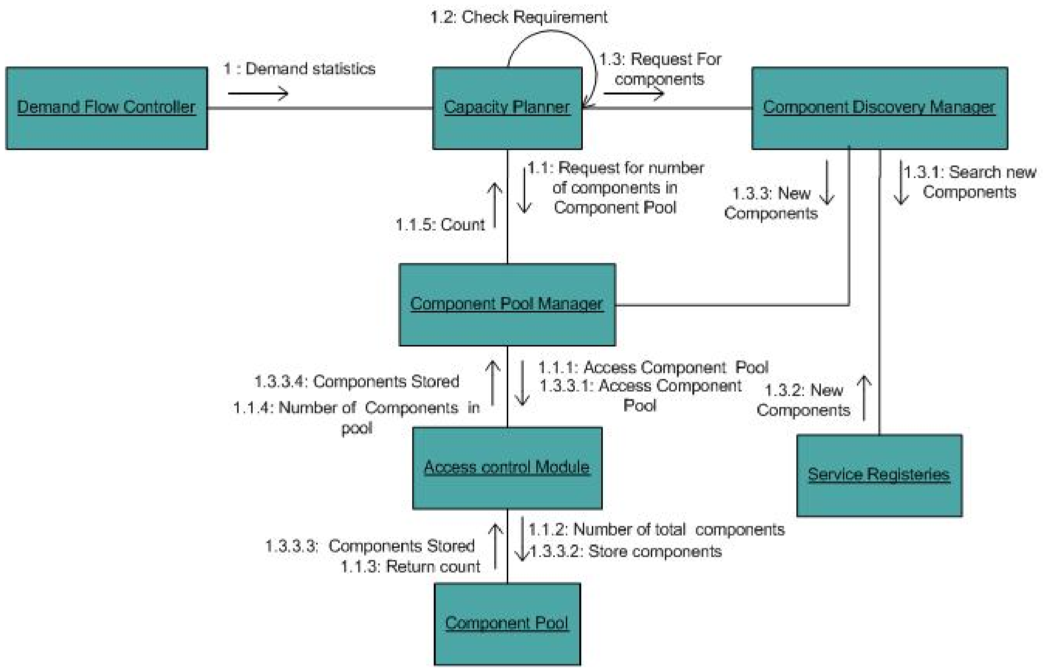

1.2: Check Requirement
1 : Demand statistics
1.3: Request For components
Demand Flow Controller
Capacity Planner
Component Discovery Manager
1.1: Request for number of components in Component Pool
1.1.5: Count
1.3.3: New Components
1.3.1: Search new Components
Component Pool Manager
1.3.3.4: Components Stored
1.1.4: Number of Components in pool
1.1.1: Access Component Pool
1.3.3.1: Access Component Pool
1.3.2: New Components
Access control Module
Service Registeries
1.3.3.3: Components Stored
1.1.3: Return count
1.1.2: Number of total components
1.3.3.2: Store components
Component Pool

Figure 4.8: Collaboration Diagram for Capacity Planning

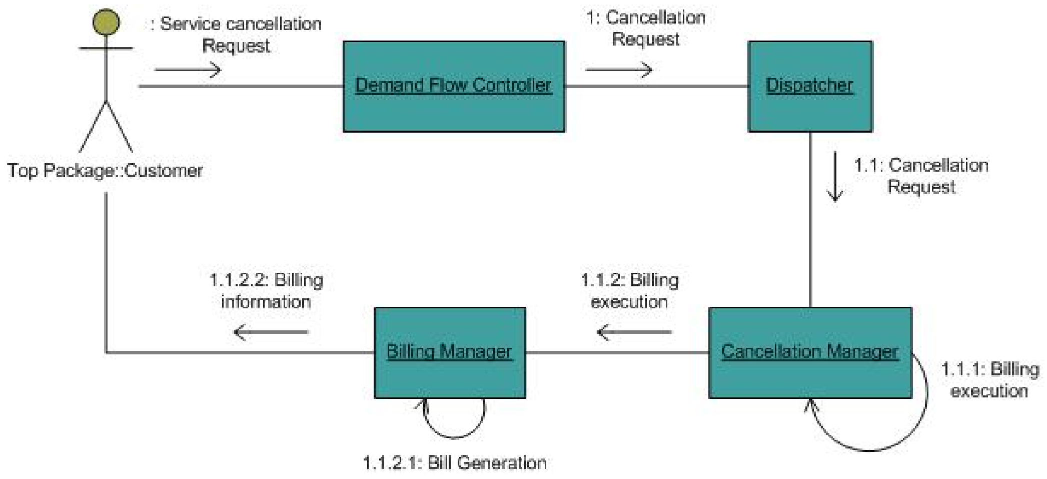


Figure 4.9: Collaboration Diagram for Cancellation and Billing Management

Table 4.1: Security Mechanisms, corresponding addressed vulnerabilities and secured CRESCENT Modules

Security Module/Mechanism	**Vulnerabilities Addressed**	**Secured Crescent Module**
Access Control Module	This module ensures that the requesting entity is authorized to access a certain resource. Only the authorized manager modules will be allowed to make changes to the resources in pools, thus eliminating the threats related to confidentiality such as masquerading, disclosure of confidential data and unauthorized accessing and tampering of sensitive data in storage pools.	SLA Manager, Workflow Pool Manager, Component Pool Manager
Authentication Module	This component will ensure the modules will mutually authenticate each other prior to any request forwarding or data exchange, to eliminate the risks such as network eavesdropping, brute force attacks, cookie replay, credential theft and risk of spoofing of modules.	All modules
Encryption Module	This module is responsible for encrypting the resources stored in SLA pool, workflow pool and components pool. This will ensure secure storage of resources and safety from data tampering attempts.	SLA pool, Workflow Pool, Component Pool.
IDMS	The modules will register with IDMS, before they perform any function related to web service composition. The credentials are checked every time, any request is received from the user, thus eliminating threats related to spoofing and masquerading.	All Modules
SSL	All communications are made secure using SSL over HTTP. This reduces session hijacking, session replay and man in the middle attacks. Also, parameter manipulation through man in the middle is also restricted.	All Modules

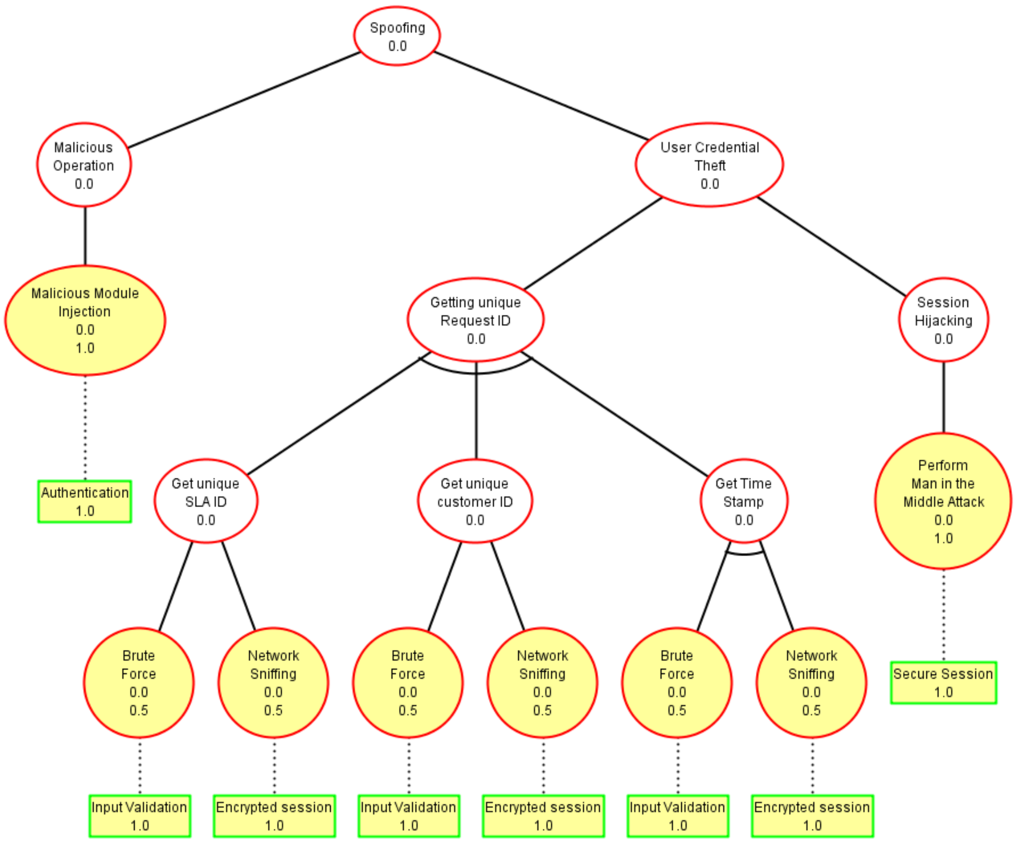


Figure 4.10: Attack prevention for spoofing threat in CRESCENT

Table 4.2: Attacks mitigated through Security Module

Security Module	**Attacks Mitigated**
Access Control Module	Denial of service, Excessive Privilege Abuse, Unauthorized privilege elevation, Injection attacks, Repudiation
Authentication Module	Network eavesdropping, Replay attacks, Credential theft
Encryption Module	Unauthorized data Alteration, Data forgery
IDMS	Login Attacks

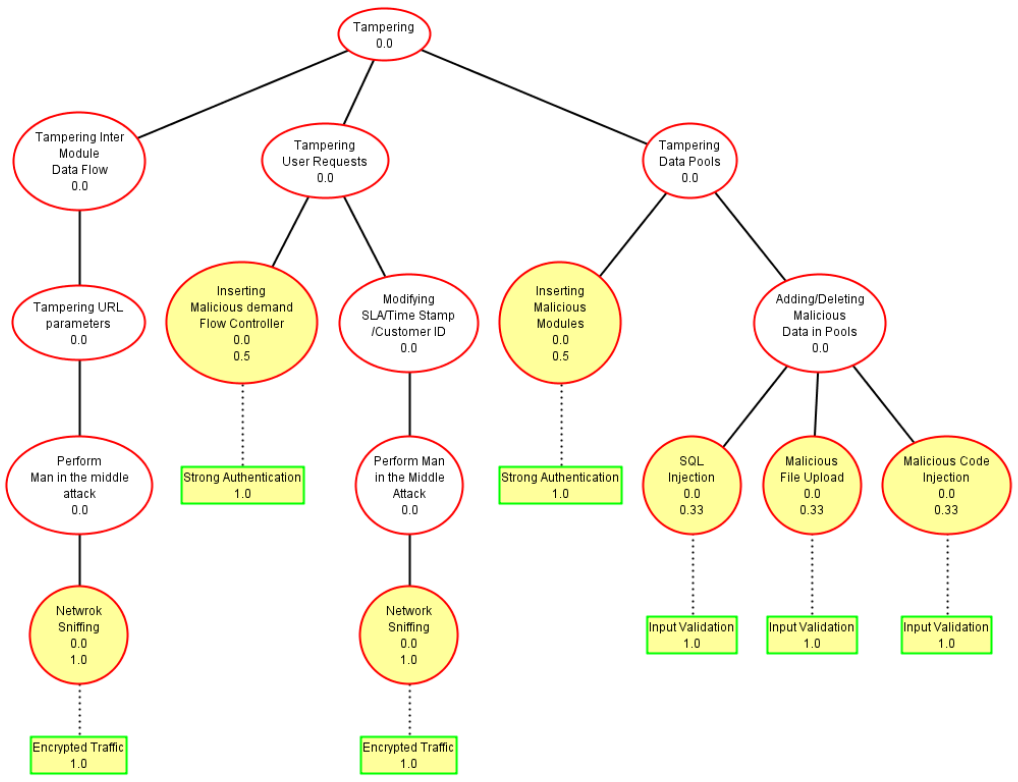


Figure 4.11: Attack prevention for Tampering threat in CRESCENT

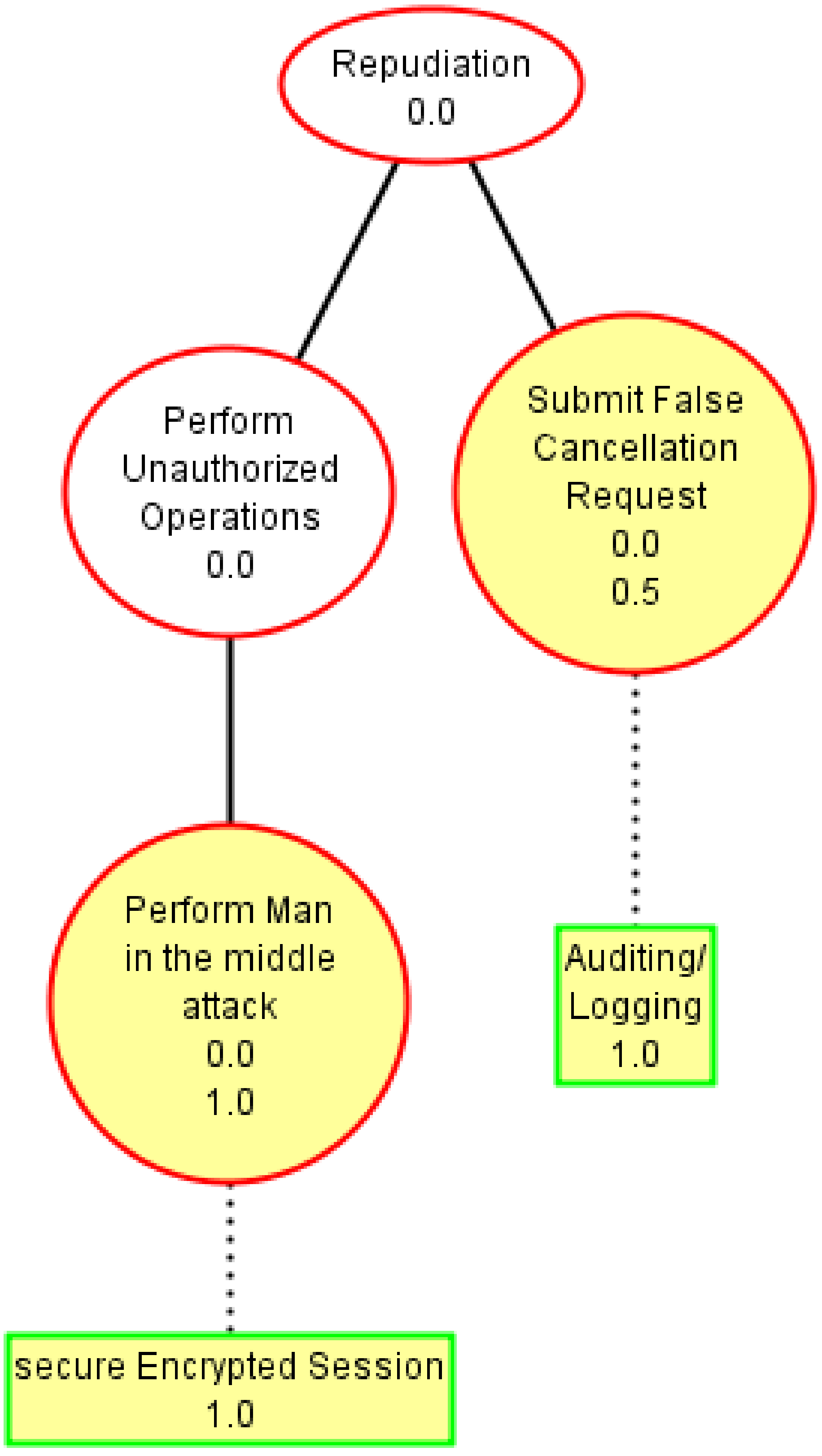

Repudiation
0.0
Perform Unauthorized Operations
0.0
Submit False Cancellation Request
0.0
0.5
Perform Man in the middle attack
0.0
1.0
Auditing/ Logging
1.0
secure Encrypted Session
1.0

Figure 4.12: Attack prevention for Repudiation threat in CRESCENT

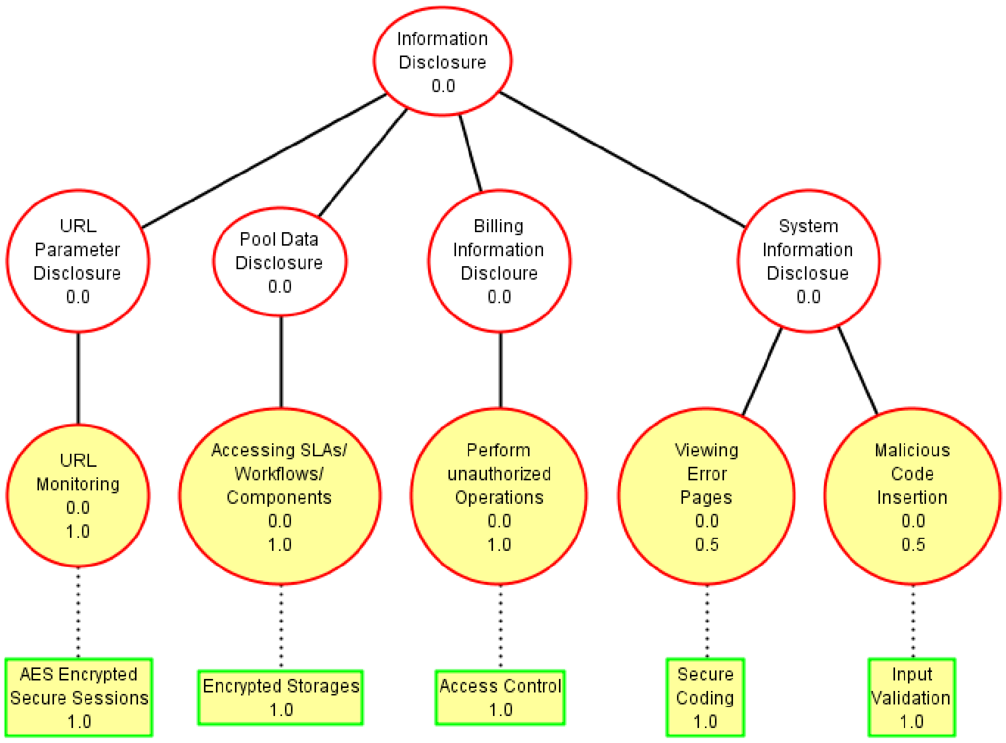

Information Disclosure 0.0
URL Parameter Disclosure 0.0
Pool Data Disclosure 0.0
Billing Information Discloure 0.0
System Information Disclosue 0.0
URL Monitoring 0.0 1.0
Accessing SLAs/ Workflows/ Components 0.0 1.0
Perform unauthorized Operations 0.0 1.0
Viewing Error Pages 0.0 0.5
Malicious Code Insertion 0.0 0.5
AES Encrypted Secure Sessions 1.0
Encrypted Storages 1.0
Access Control 1.0
Secure Coding 1.0
Input Validation 1.0

Figure 4.13: Attack prevention for Information Disclosure in CRESCENT

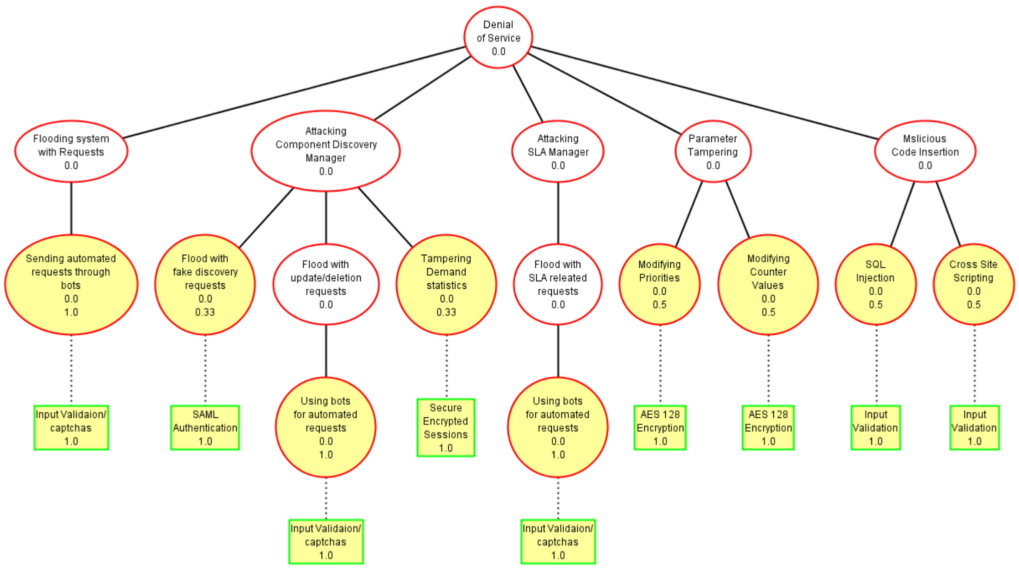

Denial of Service 0.0
Flooding system with Requests 0.0
Attacking Component Discovery Manager 0.0
Attacking SLA Manager 0.0
Parameter Tampering 0.0
Mslicious Code Insertion 0.0
Sending automated requests through bots 0.0 1.0
Flood with fake discovery requests 0.0 0.33
Flood with update/deletion requests 0.0
Tampering Demand statistics 0.0 0.33
Flood with SLA releated requests 0.0
Modifying Priorities 0.0 0.5
Modifying Counter Values 0.0 0.5
SQL Injection 0.0 0.5
Cross Site Scripting 0.0 0.5
Input Validaion/ captchas 1.0
SAML Authentication 1.0
Using bots for automated requests 0.0 1.0
Secure Encrypted Sessions 1.0
Using bots for automated requests 0.0 1.0
AES 128 Encryption 1.0
AES 128 Encryption 1.0
Input Validation 1.0
Input Validation 1.0
Input Validaion/ captchas 1.0
Input Validaion/ captchas 1.0

Figure 4.14: Attack prevention for Denial of Service in CRESCENT

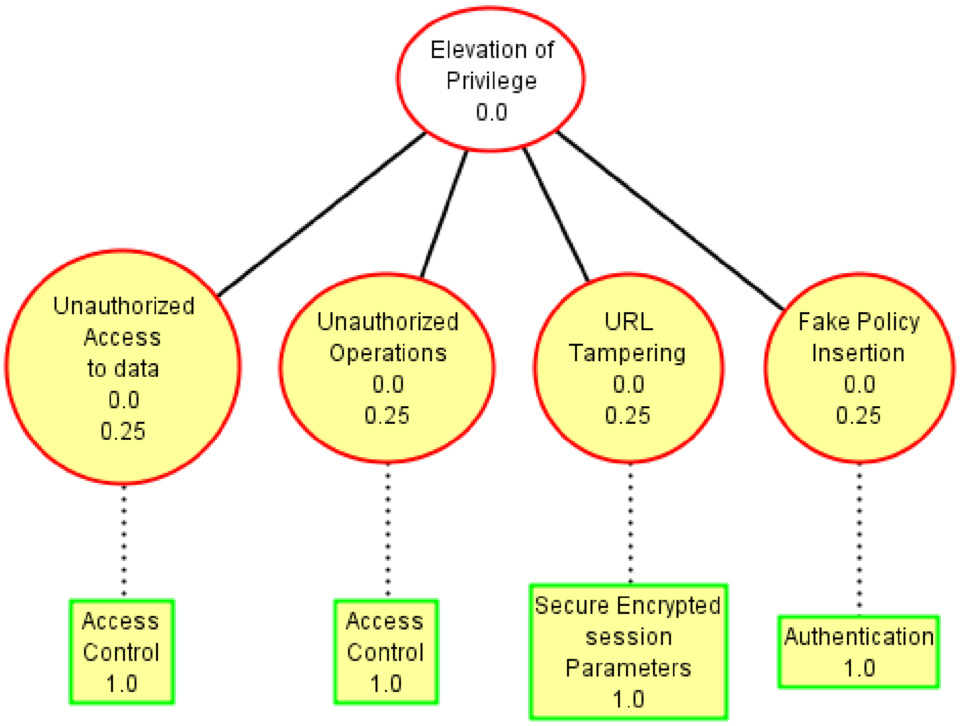

Elevation of Privilege 0.0
Unauthorized Access to data 0.0 0.25
Unauthorized Operations 0.0 0.25
URL Tampering 0.0 0.25
Fake Policy Insertion 0.0 0.25
Access Control 1.0
Access Control 1.0
Secure Encrypted session Parameters 1.0
Authentication 1.0

Figure 4.15: Attack prevention for Elevation of Privilege in CRESCENT

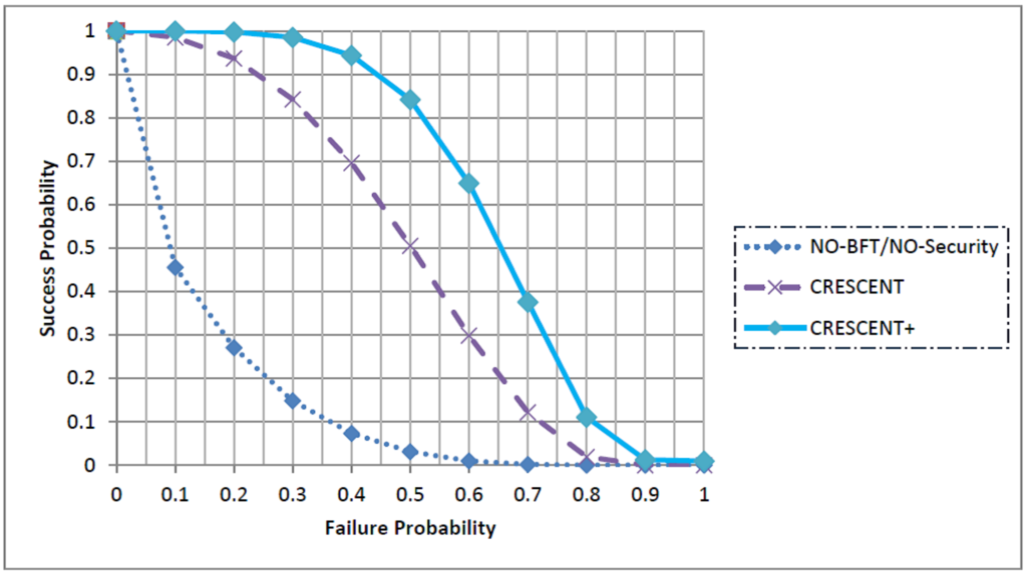


Figure 4.16: Results of Reliability Comparison

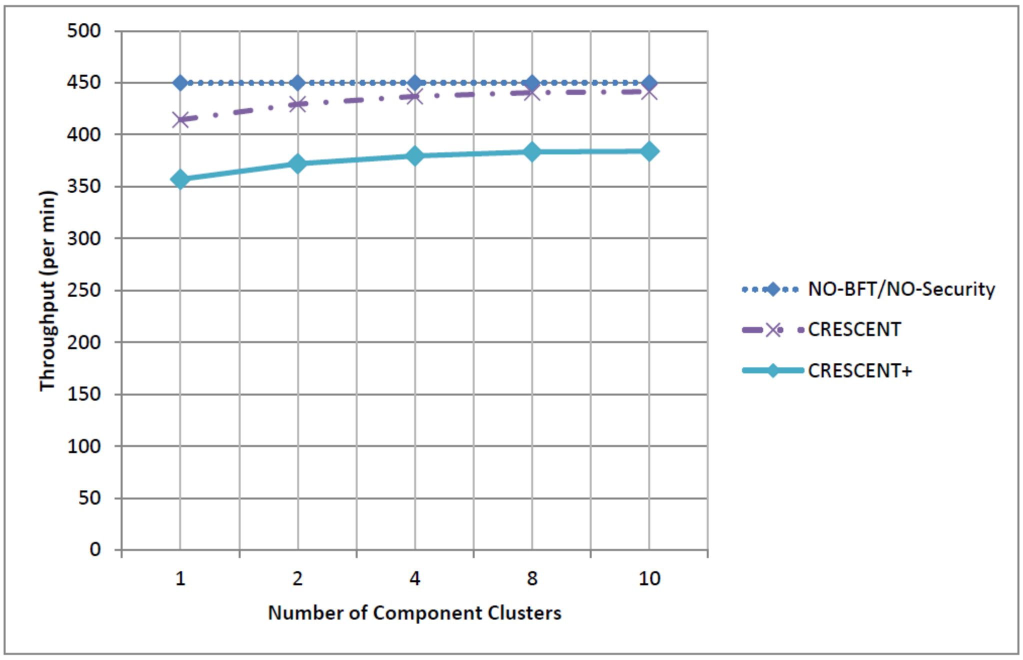


Figure 4.17: Performance Comparison when $\lambda = 30$

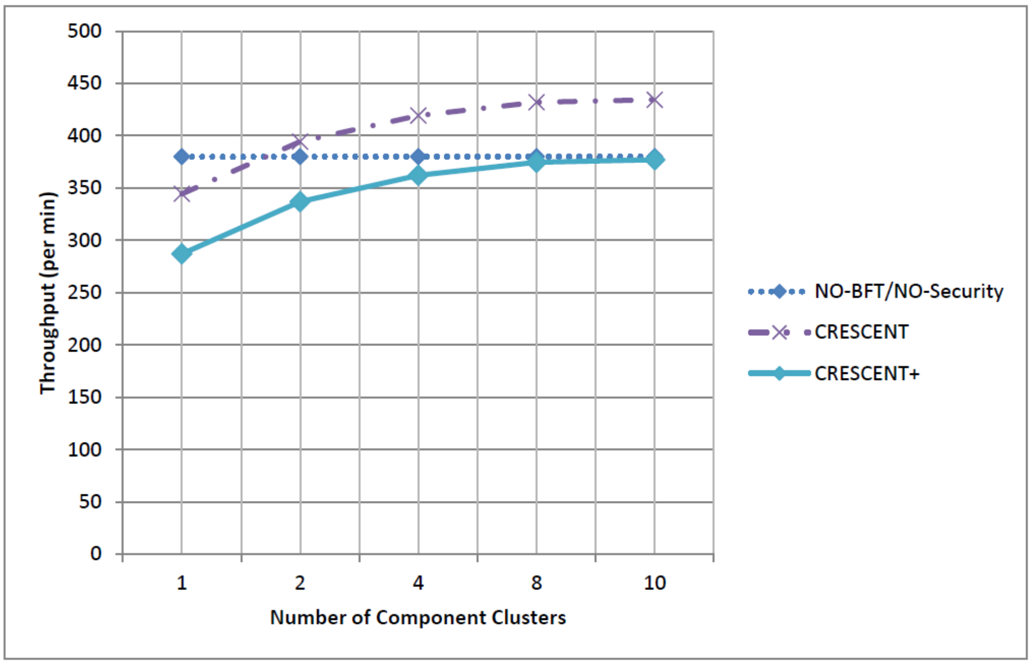


Figure 4.18: Performance Comparison when $\lambda = 100$

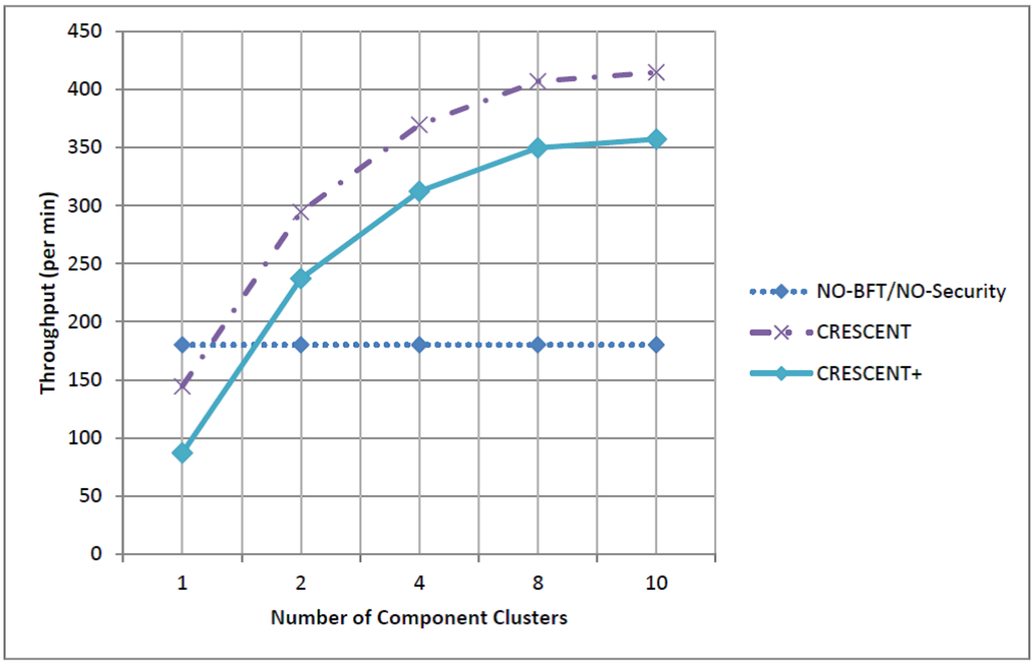


Figure 4.19: Performance Comparison when $\lambda = 300$

Chapter 5

Design and Implementation

This chapter includes the details regarding the implementation done in accordance with the research. Owing to the vast scope of the research, we have taken out a sub module for the sake of limiting the overall scope. The implementation part includes creation of a secure session for securing the communication between the client and server while accessing a composite web service. Here we have focused on providing authentication to users and a secure channel for credentials, input and output exchange between client and server for registering, logging in and usage of the web service.

5.1 Introduction

Our proposed framework incorporates a number of security features for provisioning of thorough and holistic framework for secure composite web service management. The framework in its entirety is very comprehensive and thus requires complete implementation for technical evaluation of every aspect, which is beyond the scope of this research work. We have segregated the security of data during communication between user and the framework; similar secure communication procedure will be applied between modules too. We have therefore designed and implemented a secure session after authentication for secure communication in our proposed framework. The details of the module are described in the next section.

5.2 Secure Auhentication Module Design

This portion contains details of the proposed scheme for implementing the module which is responsible for providing the secure channel for secure client server communication for availing the services. Also, another idea behind

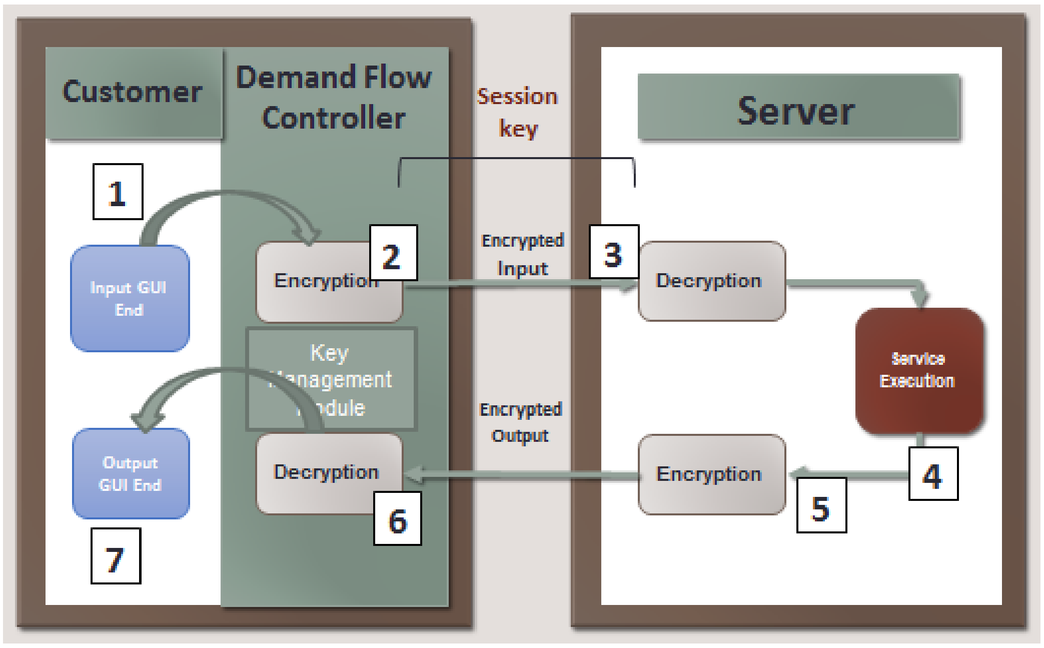


Figure 5.1: High Level Architecture of Design

the implementation is to remove any information specific to the session as soon as the user avails the service and logs out.

5.2.1 High level Architecture:

The basic high level architecture followed in designing of scheme is shown in the following diagram:

Overview:

Pre Communication: The client communicates with the server whenever the service is to be availed. Before any communication starts, RSA (public and private) keys for server are generated. The client then registers itself with the server. The registration credentials are sent to server and stored in MySQL database. ***During Communication:*** After signing in, the client generates a session key (AES) and encrypts it with servers public key. The session key is exchanged with the server and the session is created. The further communication is done using this session key which is specific to the session of the client. ***Post Communication:*** When the client has interacted with the server and utilized the service, client logs out, during which the server keys and the session key is destroyed and a new session key is generated, the next time the user logs in.

5.2.2 Implementation Environment:

Following are the details regarding implemented module of the proposed framework.
Tools and technologies The technologies employed in implementing the design of the sub module are as follows:

1. Php language and libraries (Crypt, File, Net, Math, and System) for implementing the ip-location mapping service at Server end.
2. MySQL database for storing the client server and session keys as well as user credentials in encrypted format.
3. IP-location mapping API for service creation.
4. OpenEsb IDE 2.3.1 (extended Netbeans) for creation of a service oriented application.

5.2.3 Use case scenario with Implementation Details:

The module that we have implemented as our research implementation, includes the portion of our proposed framework, where a user interacts with the framework, registers itself and avails a service in a secure manner. For this purpose, we have created a sub module that uses RSA and AES algorithm for creation of public and private keys of server and client as well as session keys specific for every user that logs in after registration.

1. Key Generation and Client Server Connection Establishment:

Before registration, server keys are generated, that are later used to securely exchange the session keys between server and the client for any specific session. After key generation, connection is established Appendix A.1 with the server at local host URL with assigned port in WAMPP (in our case we set it as localhost: 81).

2. Registration:

After connection establishment (See A.2), the user is prompted to enter its login credentials (See Appendix B.1), if he is already registered. The credentials are taken by Demand Flow Controller. But the first time users need to register prior to logging in. The user first enters his Username, Email Address and Password for sake of registration (See Appendix B.2, B.3, B.4). The credentials are encrypted, encoded and password is salt hashed and sent to server, where they are stored in MySQL DB. On successful registration, the user is prompted to log in (See Appendix B.6).

Log In:

The user after registration is prompted to log in and asked for his Email and Password. The session key is created using AES algorithm and saved in DB for the whole session in the database table, session (See Appendix C.2). All messages are exchanged after encryption with this key. The encrypted password after hashing and the email is sent to server, and decrypted there. The credentials are checked in the database and eventually the user is authenticated. As soon as the user is authenticated, the user is notified of his secure session and allowed an interface to provide input for the available service. (See Appendix C.1)

Web Service Call:

The user enters an IP address (See Appendix C.3), whose location he needs to figure out. The input data is encrypted with session key and sent to server, where it is decrypted with the same session key (stored in database) and the output is generated using IP to location providing API. The output containing the City Name and Country is generated and encrypted with the session key. The output is sent to client and is displayed at client end (See Appendix C.4). During web service invocation, the user activity is also logged, for future auditing reference.

Log Out:

After the service is availed, the user can log out at any time. After logging out the users session is destroyed and the session key and server key pair is automatically deleted, thus the session key and server key pair for certain user has life time time of one session. Next time, when the same user logs in, a new session key and server key pair is generated for the session.

The code snippets are attached at Appendix E for reference.

5.3 Discussion

Now days, web services are being very commonly used as a very efficient and cost effective way to achieve any functionality. Various types of web services are used in varying types of environments based on requirement of organization or users that are using them. Utilization of specific web service requires a number of considerations depending on functional and non-functional need of users. Different types of web services assist the service providers in providing customers with their desired functionality with desired constraints.
Rest and SOAP are most commonly known types of web services. Both have their own specific ways for composition through orchestration and other mechanisms. SOAP services transmit requests and responses in XML format encapsulated in SOAP body with SOAP headers, while REST services can use XML or JSON formats for requests and responses. In spite of the fact that both types have their own advantages and disadvantages, we have designed and utilized JSON format through stateless REST services for just providing proof of concept for the whole framework design, as JSON format is comparatively easier to parse and manage the requests and responses to/from the server side. Although this should be kept in mind that using REST for implementing prototype does not infer that SOAP is not usable with this framework. The choice of using REST or SOAP lies entirely with the system administrator of the organization utilizing the framework, who will decide which to choose with general changes based on network configurations and other systems of organization and what technologies are already in place in the organization.

5.3.1 *Concept of Digital Envelope:*

Symmetric and public key cryptography are commonly used encryption techniques, both having their own advantages and disadvantages. Symmetric key algorithms although speedy but involve risk of key leakage while being shared over the network. Public key cryptography is considered to be the most effective way of key sharing between two entities as private key of any entity s never shared over the network. But public key cryptography has low speed and takes more time than symmetric key cryptography as a lot of computation is involved in generation of keys. It is considered to be a good practice to use both symmetric and public key encryption at the same time. Therefore, we have used the concept of digital envelope which has been employed to provide *authentication* to service users. Both of the following techniques together comprise Digital Envelope methodology.

1. The public key cryptography is used to generate public-private key pair

for user authentication while ensuring non-repudiation of users

2. The service input data and output data is encrypted using shared symmetric 128 bit key generated using AES algorithm

The implemented module is prototype alias of Demand Flow Controller of CRESCENT+ that is responsible for receiving user requests for the service utilization. Following security features have also been incorporated:

1. For sake of preventing replay attacks, the user data being shared over the network for authentication is encrypted using servers public key.
2. Every time a user logs in, a new session key is created which is immediately destroyed as soon as the user logs out.
3. Server public key pair generated in start lasts for only one user session and is generated every time a new user logs in to utilize the service.

The designed prototype is just for proof of concept and may be enhanced based on user and service providers requirements.

Chapter 6

Conclusion

Composite web services are a recent advancement in the domain of web applications. A lot of work has been done focusing on the security of composite web services but no secure framework for management of composite web services has been proposed yet. This thesis document summarized the CRESCENT framework and then addressed different security aspects regarding the framework. Contribution of this research work is four-fold as shown in Figure 6.1. First, we analyzed and compared various frameworks for composite web services and identified their security loopholes. Later, we identified the security requirements of CRESCENT framework that need to be incorporated. We then performed threat modeling on CRESCENT, to analyze possible threats to the framework from adversary's perspective, using STRIDE. After studying various security vulnerabilities of the framework, we have introduced different security solutions for every vulnerability type.

In our proposed solution, CRESCENT+, we have incorporated number of security modules to enhance the security of the CRESCENT framework. The proposed architecture of CRESCENT+ has all the security requirements incorporated in it, thus providing an efficient and secure management framework for composite web services. Experimental results show CRESCENT reliability has been increased when the proposed security protocols are incorporated; however the obtained throughput values have decreased. We argued that obtained throughput values are still better when compared to existing industrial standards. The results show that Sec-CRESCENT effectively provides a reliable and secure way for web service composition over dynamic internet systems such as cloud computing. Moreover, it allows the user to fully benefit from the process of web service composition with a guarantee of security, reliability and quality of service.
Our research work has varying future direction. A few are given below:

1. The research work could be extended in direction of actual BPEL

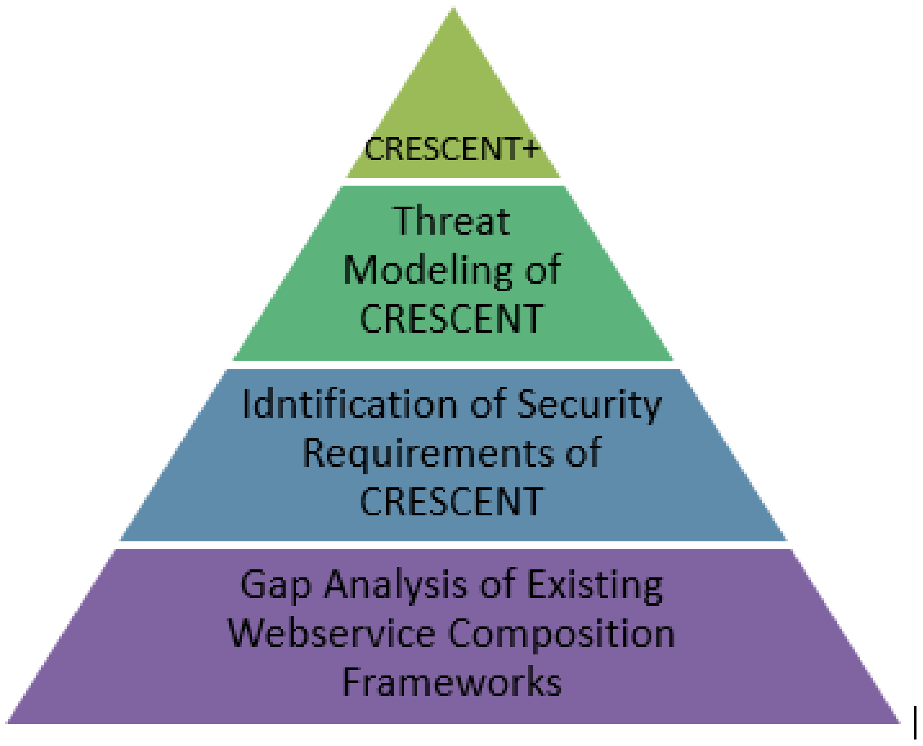
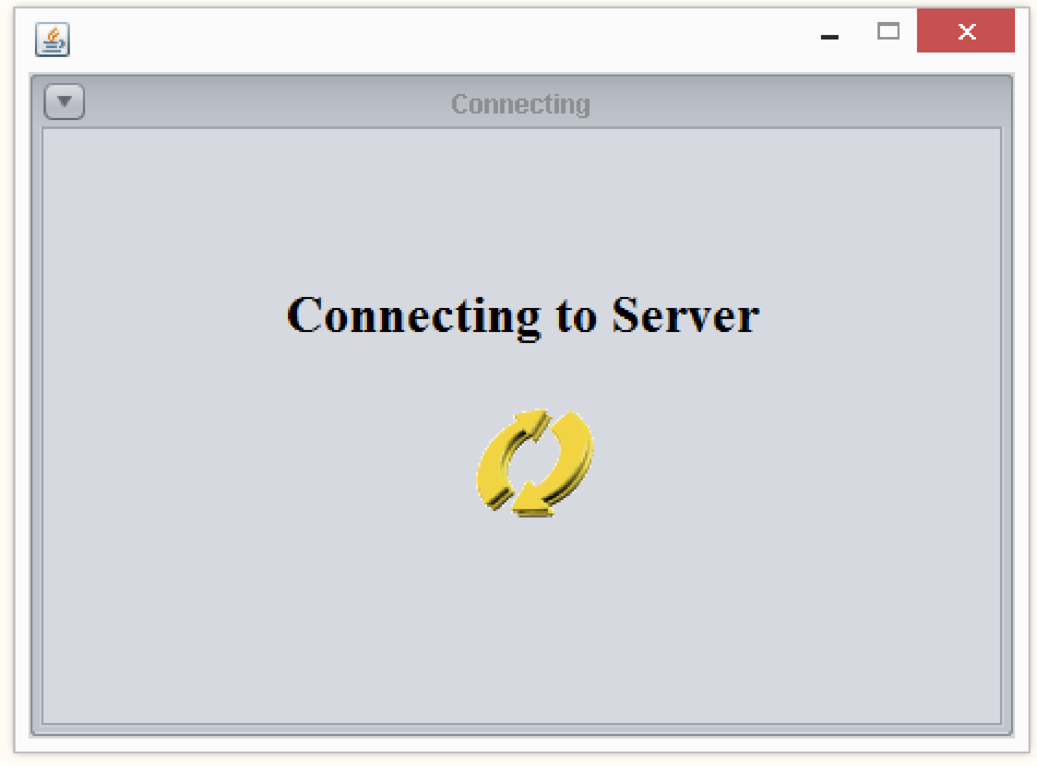

CRESCENT+
Threat Modeling of CRESCENT
Idntification of Security Requirements of CRESCENT
Gap Analysis of Existing Webservice Composition Frameworks

Figure 6.1: Four-fold Contribution of the Research

process security with respect to composite web services. The CRESCENT+ framework may be evaluated for different types of web service composition techniques. The Adaptive Composer is involved in creating BPEL scripts of composite web services which are then send to dispatcher for execution.

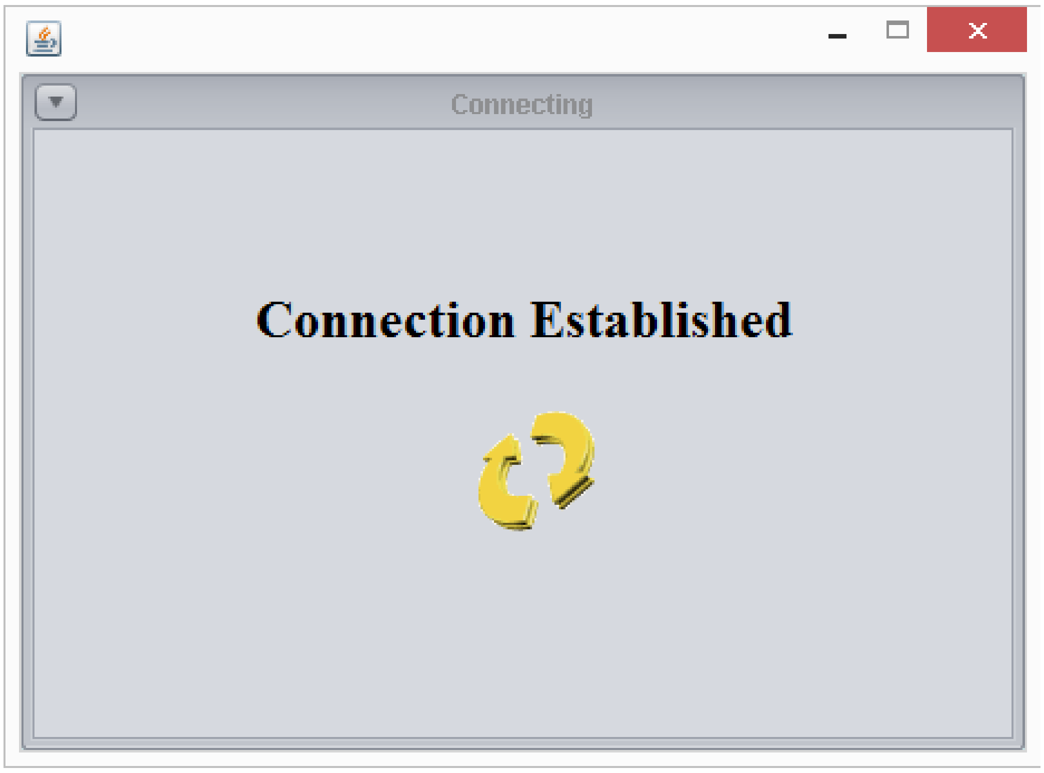
2. After incorporation of security modules, the framework could be implemented for composite web services specific environment leading to a major contribution to existing systems that provide web based services and resulting effect on performance of existing systems could be studied. Various recommendations based on real life scenario, regarding usage specific technology choices may be made.

Appendix A



Connecting to Server

Figure A.1:



Connection Established

Figure A.2:

Appendix B

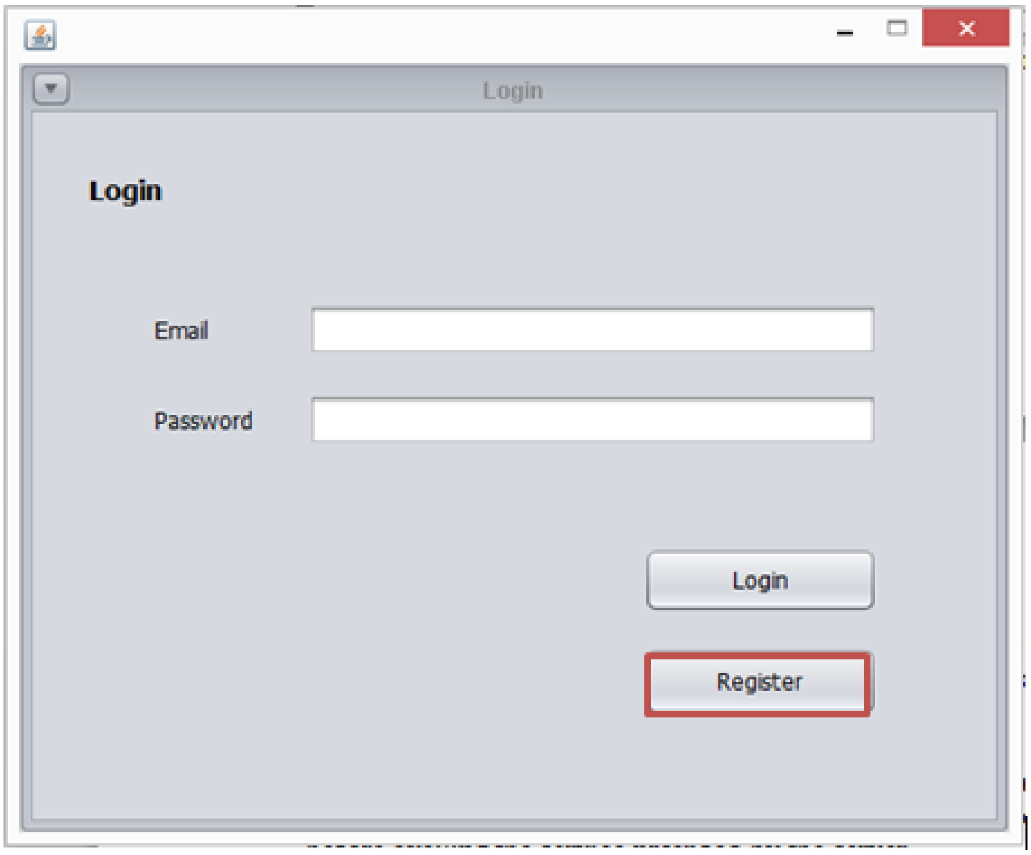


Figure B.1:

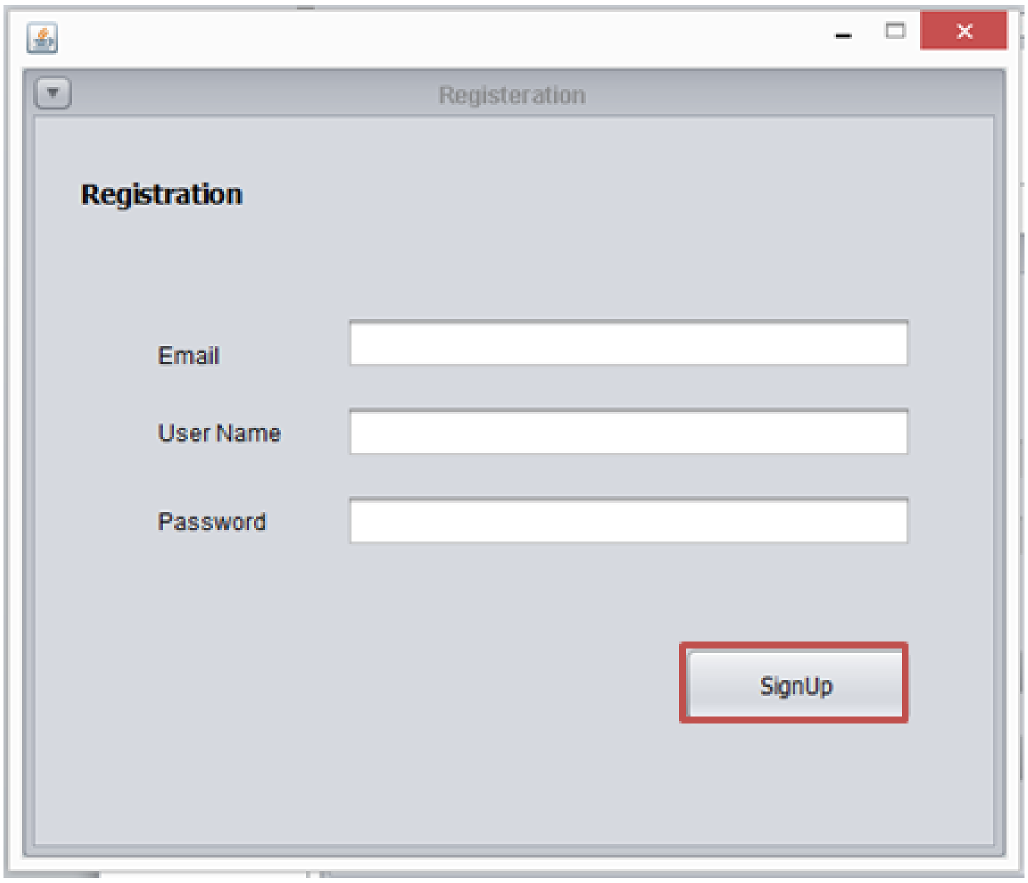
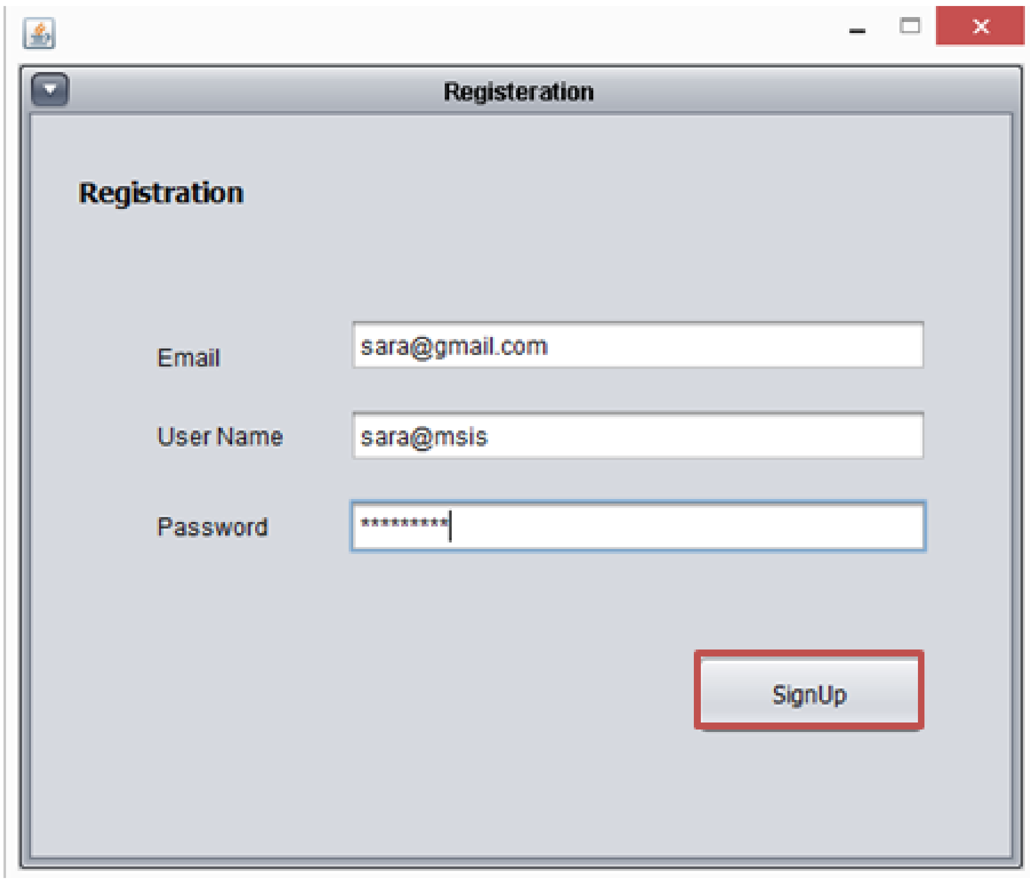

Registeration
Registration
Email
User Name
Password
SignUp

Figure B.2:



Registeration

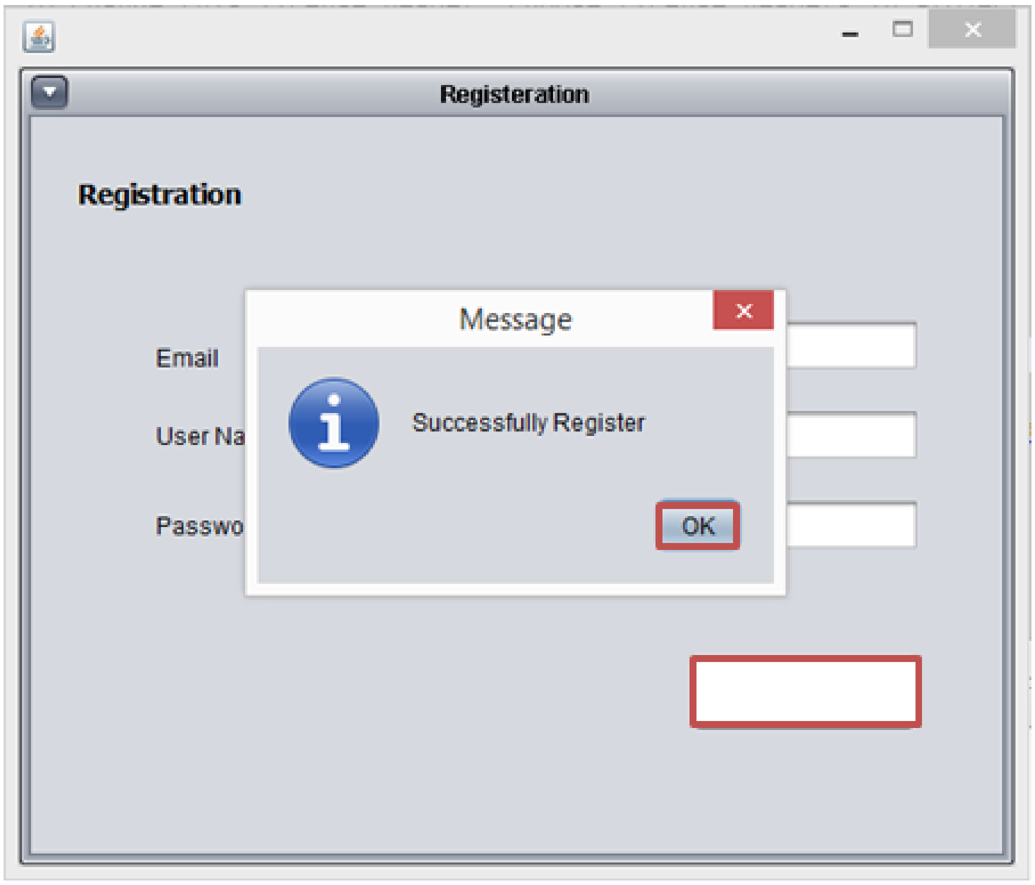
Registration

Email sara@gmail.com

User Name sara@msis

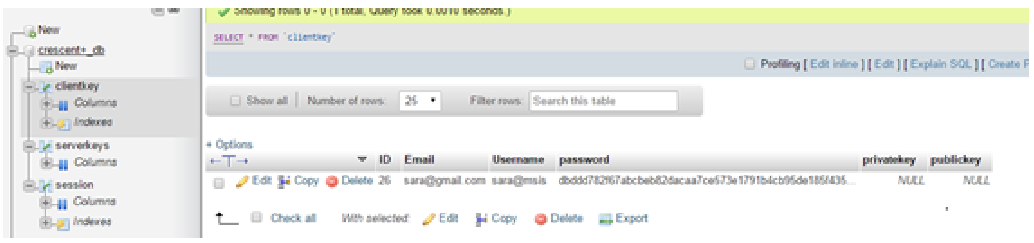
Password *********

Figure B.3:



Successfully Register

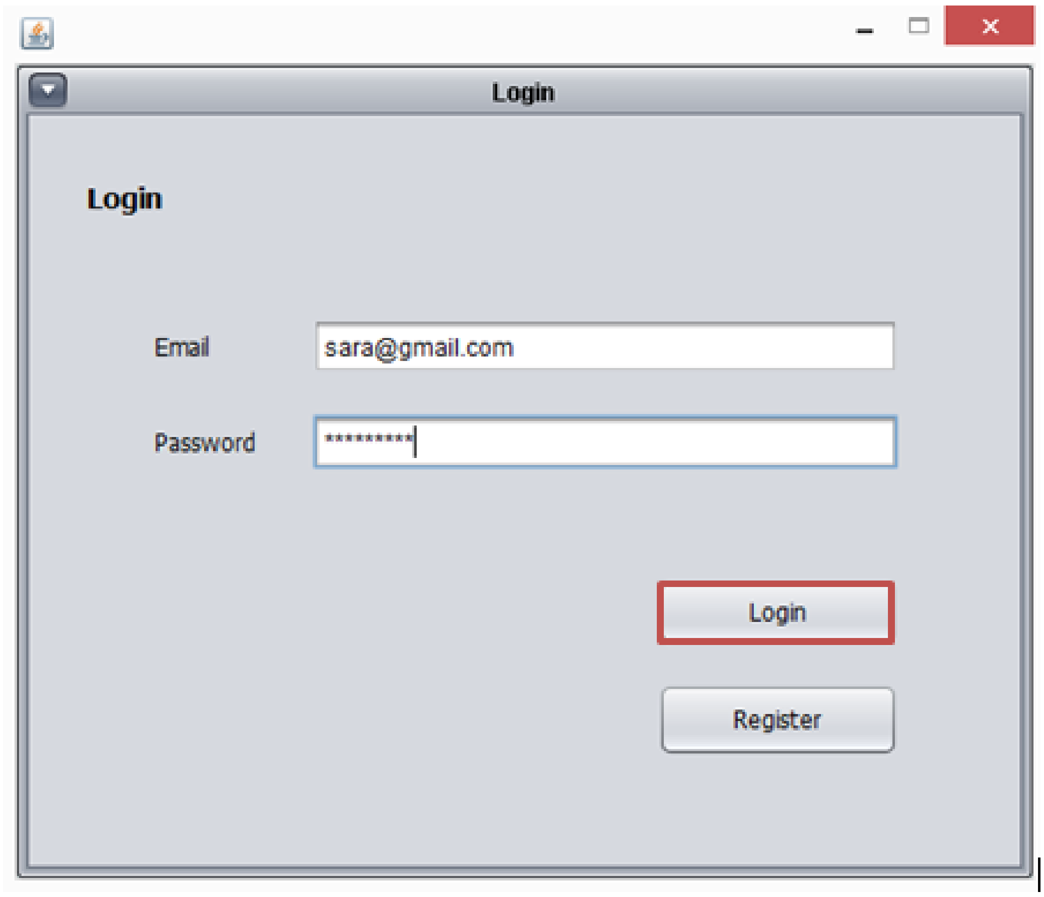
Figure B.4:


New
crescent+_db
New
clientkey
Columns
Indexes
serverkeys
Columns
session
Columns
Indexes
Profiling [Edit inline] [Edit] [Explain SQL] [Create P
Show all
Number of rows: 25
Filter rows: Search this table
+ Options
Check all
With selected: Edit Copy Delete Export

```
Showing rows 0 - 0 (1 total, Query took 0.0010 seconds.)
SELECT * FROM `clientkey`
```

	ID	Email	Username	password	privatekey	publickey
Edit Copy Delete	26	sara@gmail.com	sara@msis	dbddd782f67abcbeb82dacaa7ce573e1791b4cb95de185f435...	NULL	NULL

Figure B.5:

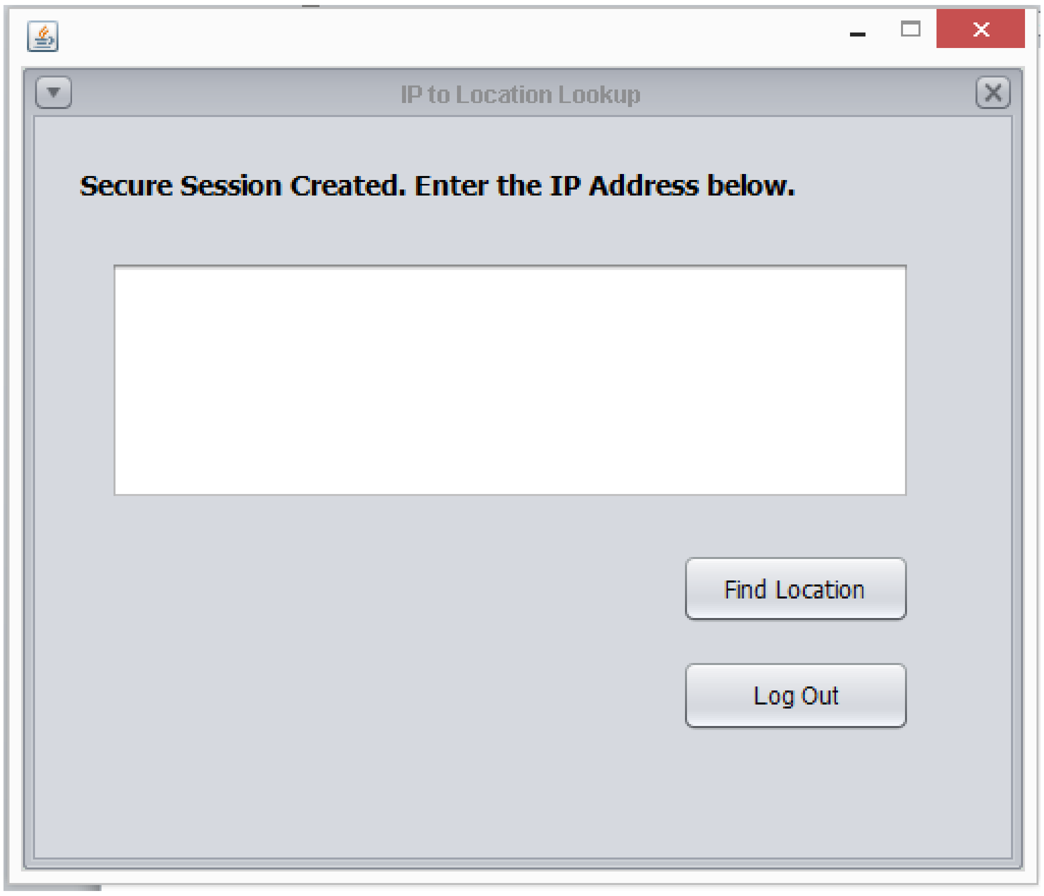


sara@gmail.com

****

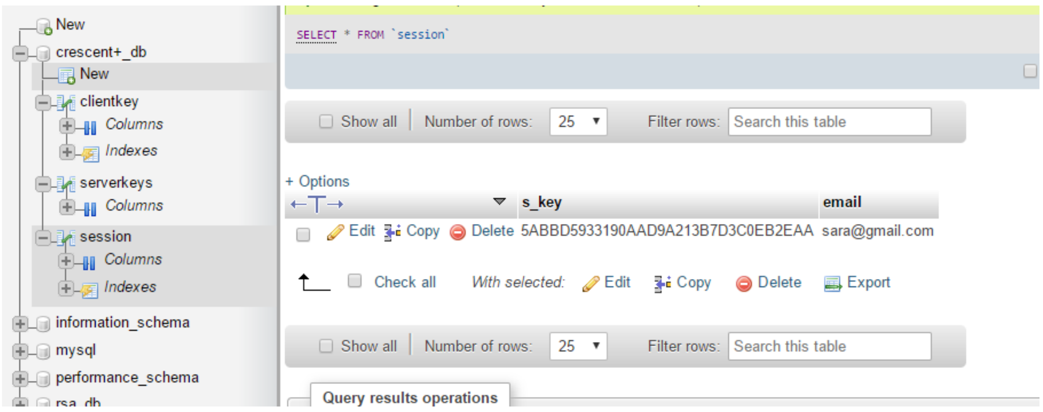
Figure B.6:

Appendix C



Secure Session Created. Enter the IP Address below.

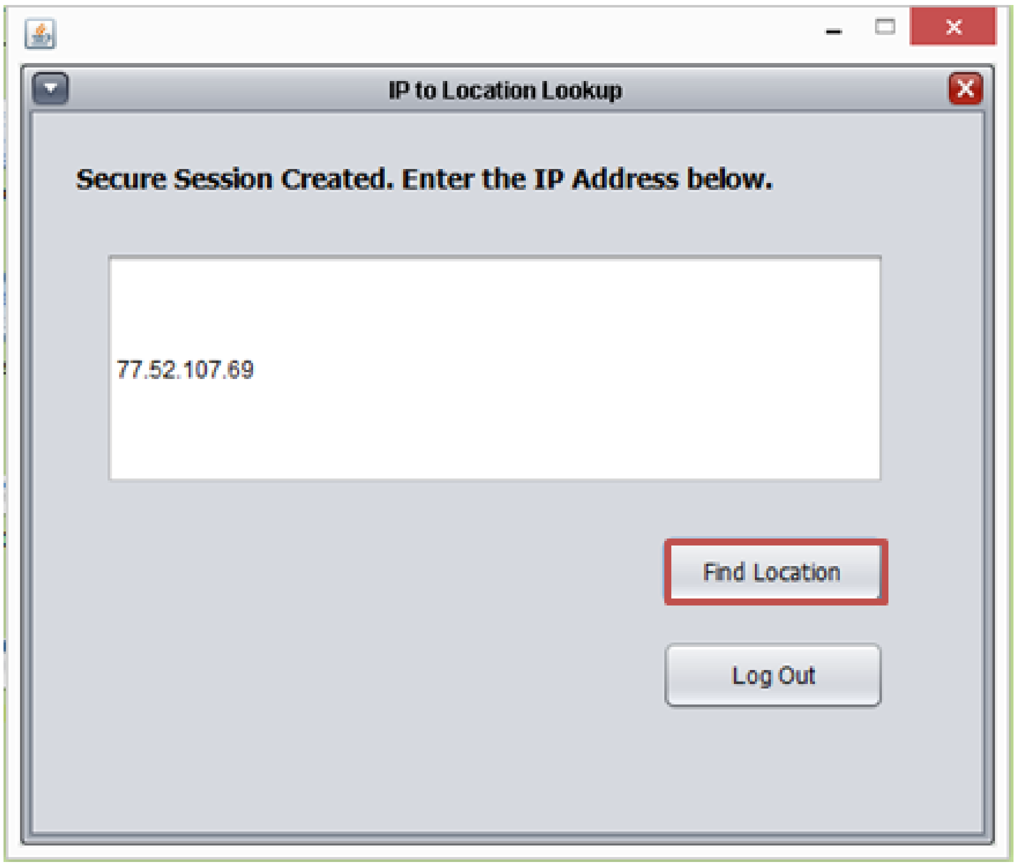
Figure C.1:



```
SELECT * FROM `session`
```

	s_key	email
Edit Copy Delete	5ABBD5933190AAD9A213B7D3C0EB2EAA	sara@gmail.com

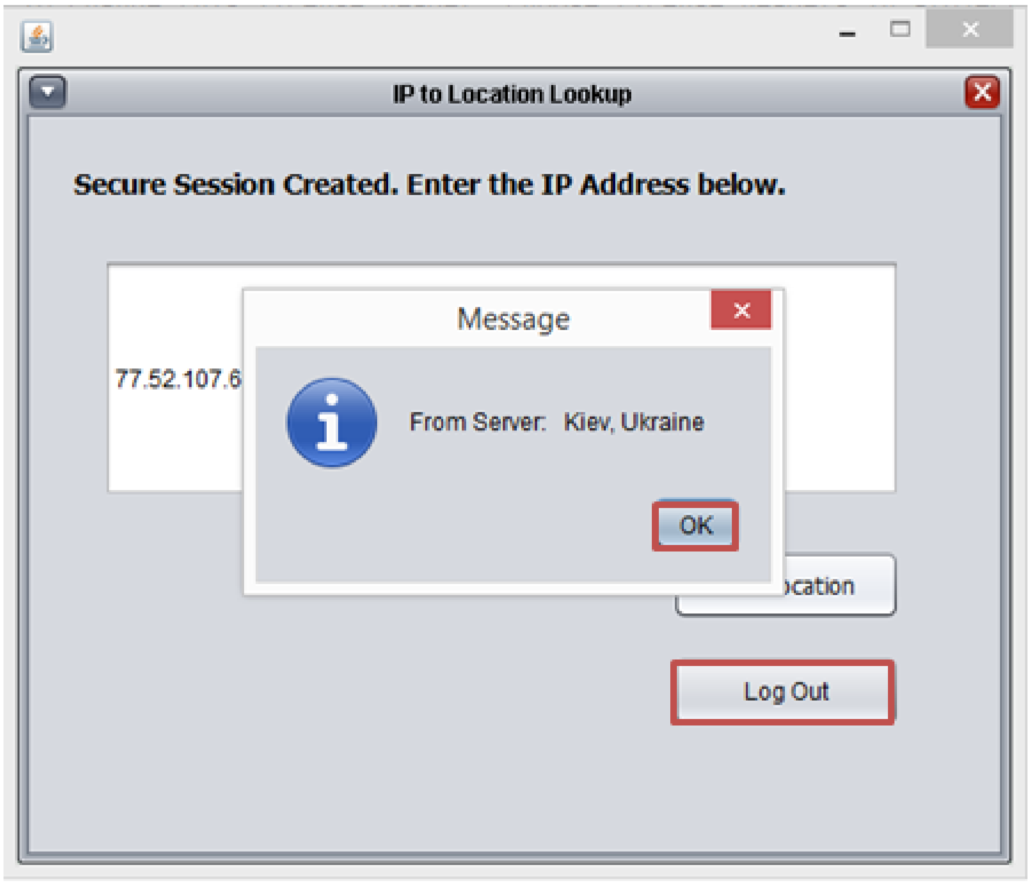
Figure C.2:



Secure Session Created. Enter the IP Address below.

77.52.107.69

Figure C.3:



Secure Session Created. Enter the IP Address below.

77.52.107.6

From Server: Kiev, Ukraine

Figure C.4:

Appendix D

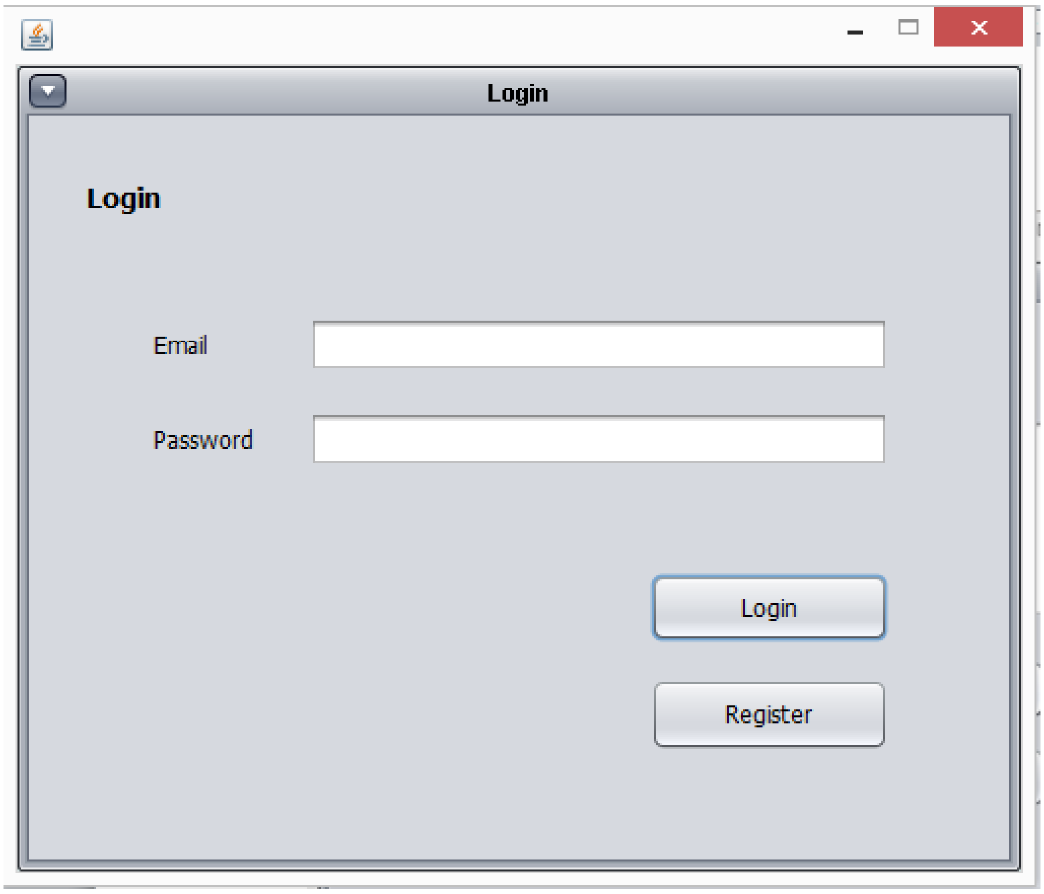
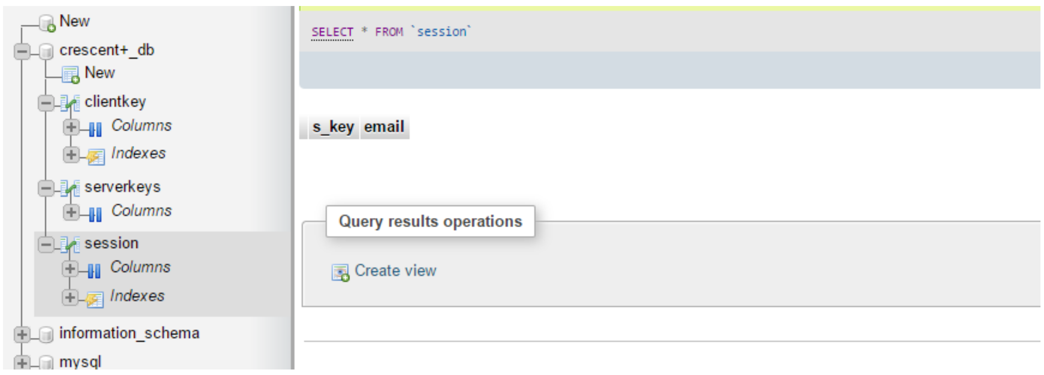


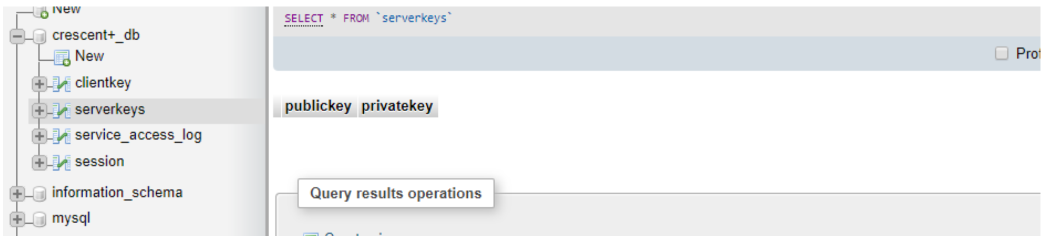
Figure D.1:



```
SELECT * FROM `session`
```

s_key	email

Figure D.2:



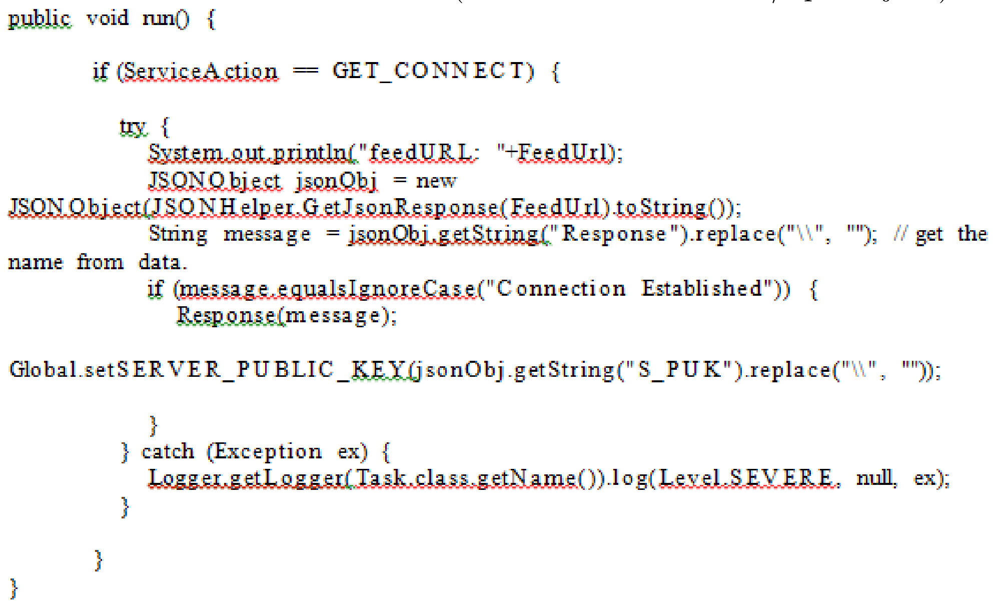
```
SELECT * FROM `serverkeys`
```

publickey	privatekey

Figure D.3:

Appendix E

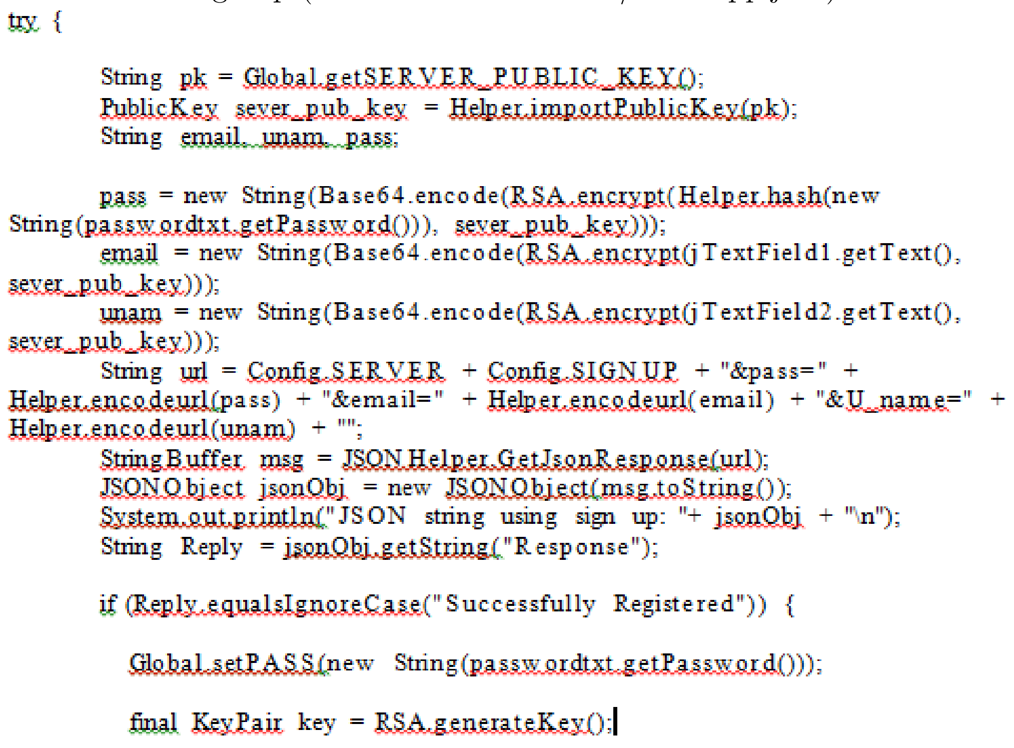
Connection Establishment (DemandFlowConroller/Splash.java)



```
public void run() {

        if (ServiceAction == GET_CONNECT) {

            try {
                System.out.println("feedURL: "+FeedUrl);
                JSONObject jsonObj = new
JSONObject(JSONHelper.GetJsonResponse(FeedUrl).toString());
                String message = jsonObj.getString("Response").replace("\\", ""); // get the
name from data.
                if (message.equalsIgnoreCase("Connection Established")) {
                    Response(message);

Global.setSERVER_PUBLIC_KEY(jsonObj.getString("S_PUK").replace("\\", ""));

                }
            } catch (Exception ex) {
                Logger.getLogger(Task.class.getName()).log(Level.SEVERE, null, ex);
            }

        }
}
```

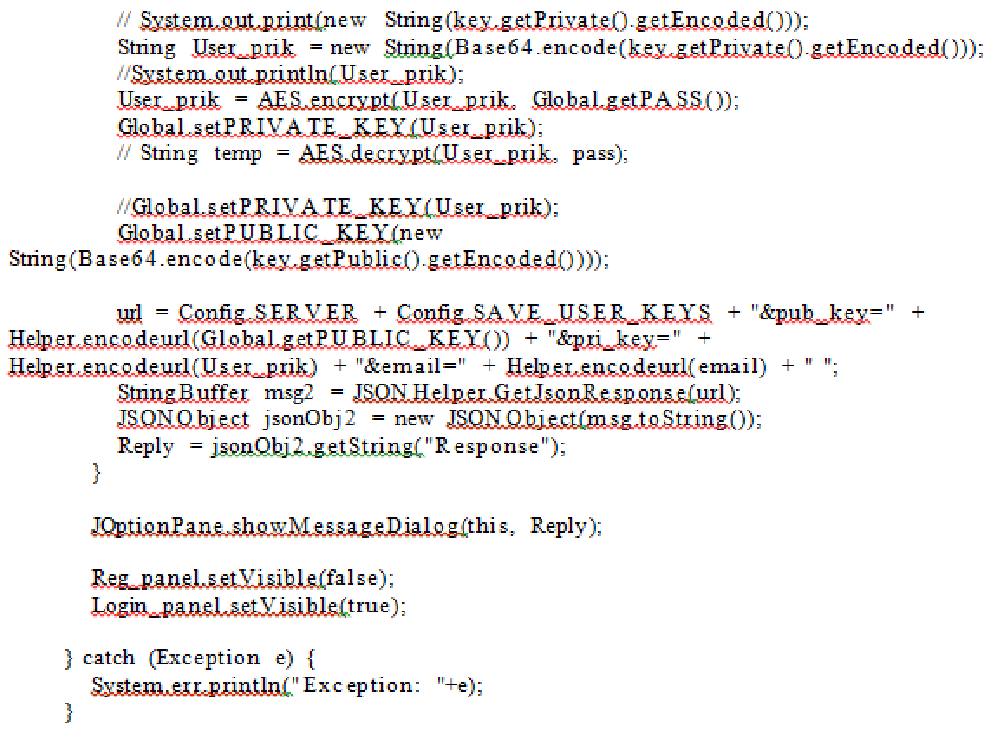
Figure E.1:

Sign up (DemandFlowConroller/StartApp.java)



```
try {

        String pk = Global.getSERVER_PUBLIC_KEY();
        PublicKey sever_pub_key = Helper.importPublicKey(pk);
        String email, unam, pass;

        pass = new String(Base64.encode(RSA.encrypt(Helper.hash(new
String(passwordtxt.getPassword())), sever_pub_key)));
        email = new String(Base64.encode(RSA.encrypt(jTextField1.getText(),
sever_pub_key)));
        unam = new String(Base64.encode(RSA.encrypt(jTextField2.getText(),
sever_pub_key)));
        String url = Config.SERVER + Config.SIGNUP + "&pass=" +
Helper.encodeurl(pass) + "&email=" + Helper.encodeurl(email) + "&U_name=" +
Helper.encodeurl(unam) + "";
        StringBuffer msg = JSONHelper.GetJsonResponse(url);
        JSONObject jsonObj = new JSONObject(msg.toString());
        System.out.println("JSON string using sign up: "+ jsonObj + "\n");
        String Reply = jsonObj.getString("Response");

        if (Reply.equalsIgnoreCase("Successfully Registered")) {

          Global.setPASS(new String(passwordtxt.getPassword()));

          final KeyPair key = RSA.generateKey();
```

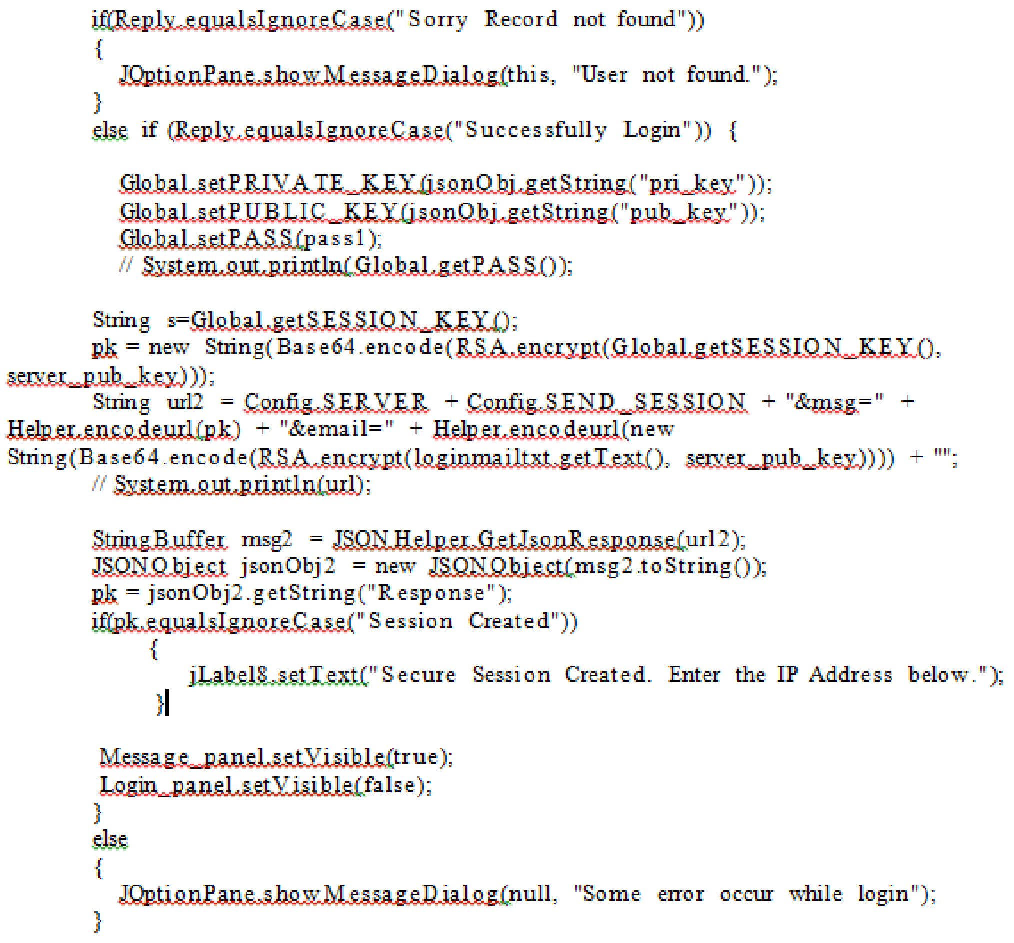
Figure E.2:



```
            // System.out.print(new String(key.getPrivate().getEncoded()));
            String User_prik = new String(Base64.encode(key.getPrivate().getEncoded()));
            //System.out.println(User_prik);
            User_prik = AES.encrypt(User_prik, Global.getPASS());
            Global.setPRIVATE_KEY(User_prik);
            // String temp = AES.decrypt(User_prik, pass);

            //Global.setPRIVATE_KEY(User_prik);
            Global.setPUBLIC_KEY(new
String(Base64.encode(key.getPublic().getEncoded())));

            url = Config.SERVER + Config.SAVE_USER_KEYS + "&pub_key=" +
Helper.encodeurl(Global.getPUBLIC_KEY()) + "&pri_key=" +
Helper.encodeurl(User_prik) + "&email=" + Helper.encodeurl(email) + " ";
            StringBuffer msg2 = JSONHelper.GetJsonResponse(url);
            JSONObject jsonObj2 = new JSONObject(msg.toString());
            Reply = jsonObj2.getString("Response");
          }

          JOptionPane.showMessageDialog(this, Reply);

          Reg_panel.setVisible(false);
          Login_panel.setVisible(true);

      } catch (Exception e) {
          System.err.println("Exception: "+e);
      }
```

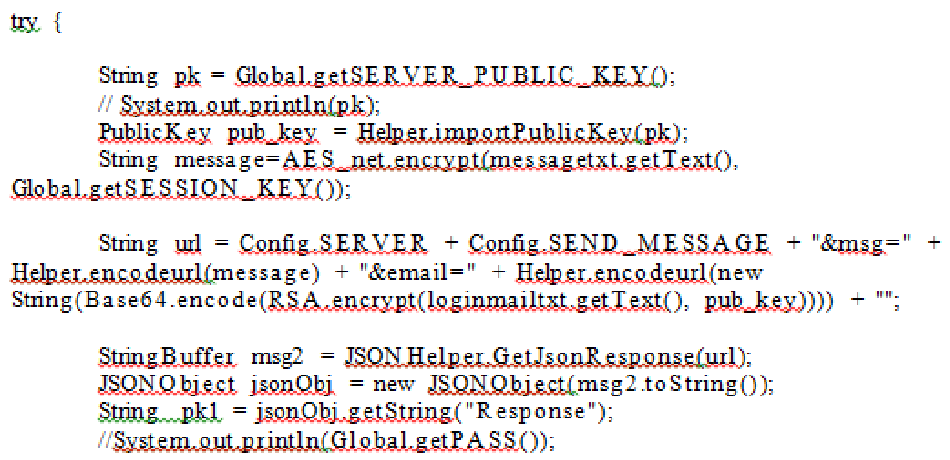
Figure E.3:

Sign In (DemandFlowConroller/StartApp.java)



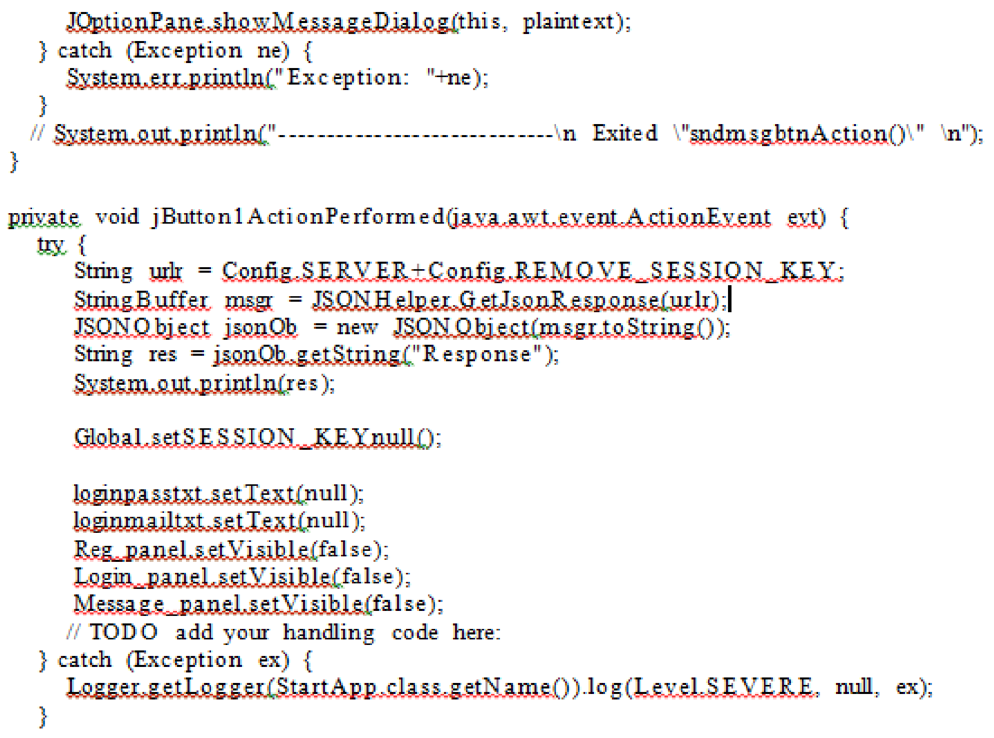
```
        if(Reply.equalsIgnoreCase("Sorry Record not found"))
        {
          JOptionPane.showMessageDialog(this, "User not found.");
        }
        else if (Reply.equalsIgnoreCase("Successfully Login")) {

          Global.setPRIVATE_KEY(jsonObj.getString("pri_key"));
          Global.setPUBLIC_KEY(jsonObj.getString("pub_key"));
          Global.setPASS(pass1);
          // System.out.println(Global.getPASS());

        String s=Global.getSESSION_KEY();
        pk = new String(Base64.encode(RSA.encrypt(Global.getSESSION_KEY(),
server_pub_key)));
        String url2 = Config.SERVER + Config.SEND_SESSION + "&msg=" +
Helper.encodeurl(pk) + "&email=" + Helper.encodeurl(new
String(Base64.encode(RSA.encrypt(loginmailtxt.getText(), server_pub_key)))) + "";
        // System.out.println(url);

        StringBuffer msg2 = JSONHelper.GetJsonResponse(url2);
        JSONObject jsonObj2 = new JSONObject(msg2.toString());
        pk = jsonObj2.getString("Response");
        if(pk.equalsIgnoreCase("Session Created"))
              {
                  jLabel8.setText("Secure Session Created. Enter the IP Address below.");
              }

         Message_panel.setVisible(true);
         Login_panel.setVisible(false);
        }
        else
        {
          JOptionPane.showMessageDialog(null, "Some error occur while login");
        }
```

Figure E.4:



```
try {

        String pk = Global.getSERVER_PUBLIC_KEY();
        // System.out.println(pk);
        PublicKey pub_key = Helper.importPublicKey(pk);
        String message=AES_net.encrypt(messagetxt.getText(),
Global.getSESSION_KEY());

        String url = Config.SERVER + Config.SEND_MESSAGE + "&msg=" +
Helper.encodeurl(message) + "&email=" + Helper.encodeurl(new
String(Base64.encode(RSA.encrypt(loginmailtxt.getText(), pub_key)))) + "";

        StringBuffer msg2 = JSONHelper.GetJsonResponse(url);
        JSONObject jsonObj = new JSONObject(msg2.toString());
        String  pk1 = jsonObj.getString("Response");
        //System.out.println(Global.getPASS());
```

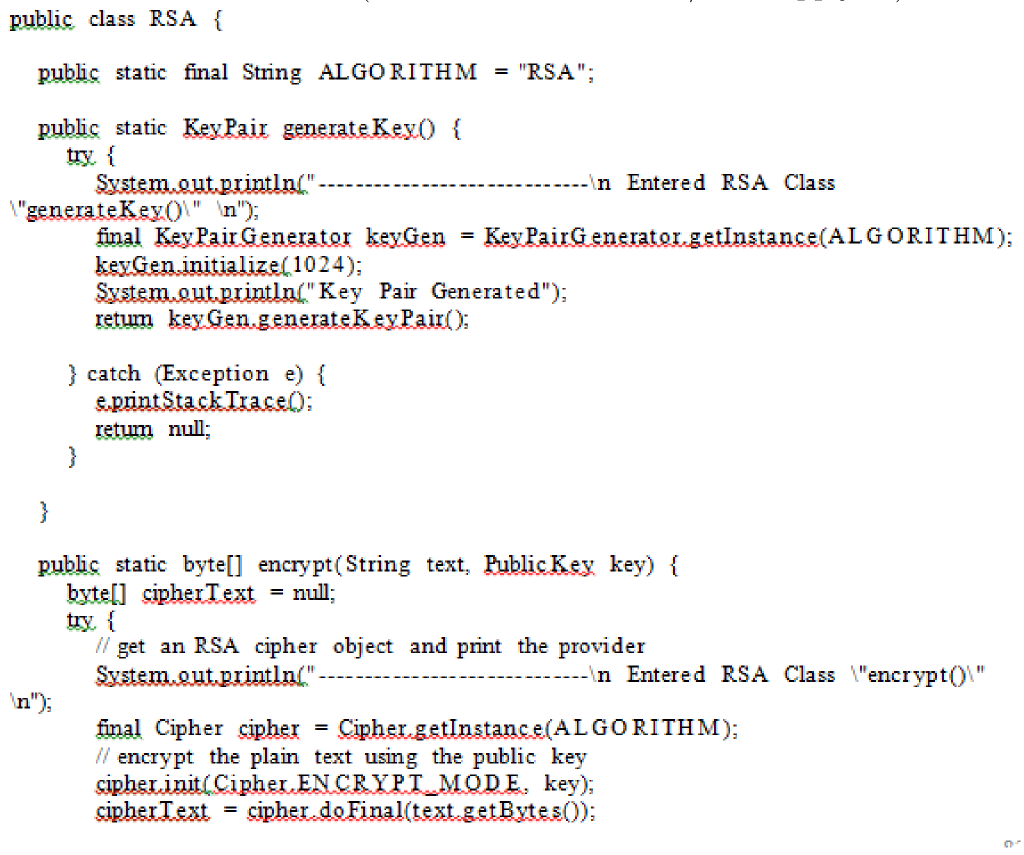
Figure E.5:



```
          JOptionPane.showMessageDialog(this, plaintext);
      } catch (Exception ne) {
          System.err.println("Exception: "+ne);
      }
     // System.out.println("----------------------------\n Exited \"sndmsgbtnAction()\" \n");
  }

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      try {
          String urlr = Config.SERVER+Config.REMOVE_SESSION_KEY;
          StringBuffer msgr = JSONHelper.GetJsonResponse(urlr);
          JSONObject jsonOb = new JSONObject(msgr.toString());
          String res = jsonOb.getString("Response");
          System.out.println(res);

          Global.setSESSION_KEYnull();

          loginpasstxt.setText(null);
          loginmailtxt.setText(null);
          Reg_panel.setVisible(false);
          Login_panel.setVisible(false);
          Message_panel.setVisible(false);
         // TODO add your handling code here:
      } catch (Exception ex) {
          Logger.getLogger(StartApp.class.getName()).log(Level.SEVERE, null, ex);
      }
```

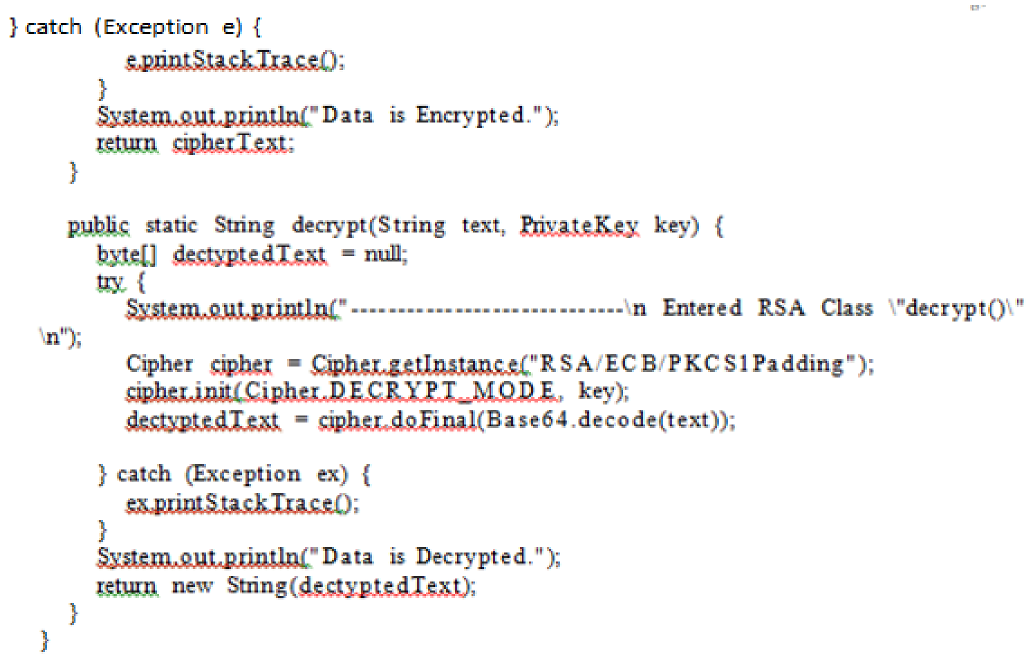
Figure E.6:

Service Utilization (DemandFlowController/StartApp.java)



```
public class RSA {

   public static final String ALGORITHM = "RSA";

   public static KeyPair generateKey() {
      try {
         System.out.println("----------------------------\n Entered RSA Class
\"generateKey()\" \n");
         final KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ALGORITHM);
         keyGen.initialize(1024);
         System.out.println("Key Pair Generated");
         return keyGen.generateKeyPair();

      } catch (Exception e) {
         e.printStackTrace();
         return null;
      }

   }

   public static byte[] encrypt(String text, PublicKey key) {
      byte[] cipherText = null;
      try {
         // get an RSA cipher object and print the provider
         System.out.println("----------------------------\n Entered RSA Class \"encrypt()\"
\n");
         final Cipher cipher = Cipher.getInstance(ALGORITHM);
         // encrypt the plain text using the public key
         cipher.init(Cipher.ENCRYPT_MODE, key);
         cipherText = cipher.doFinal(text.getBytes());
```

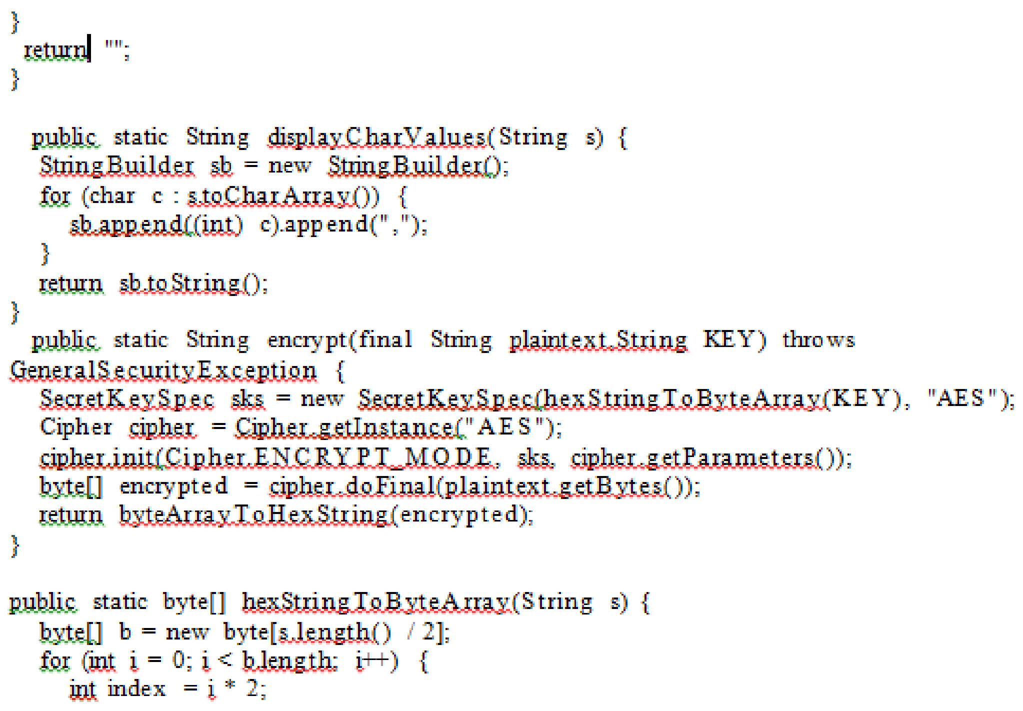
Figure E.7:



```
} catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("Data is Encrypted.");
        return cipherText;
    }

    public static String decrypt(String text, PrivateKey key) {
        byte[] dectyptedText = null;
        try {
            System.out.println("---------------------------\n Entered RSA Class \"decrypt()\"
 \n");
            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            cipher.init(Cipher.DECRYPT_MODE, key);
            dectyptedText = cipher.doFinal(Base64.decode(text));

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        System.out.println("Data is Decrypted.");
        return new String(dectyptedText);
    }
 }
```

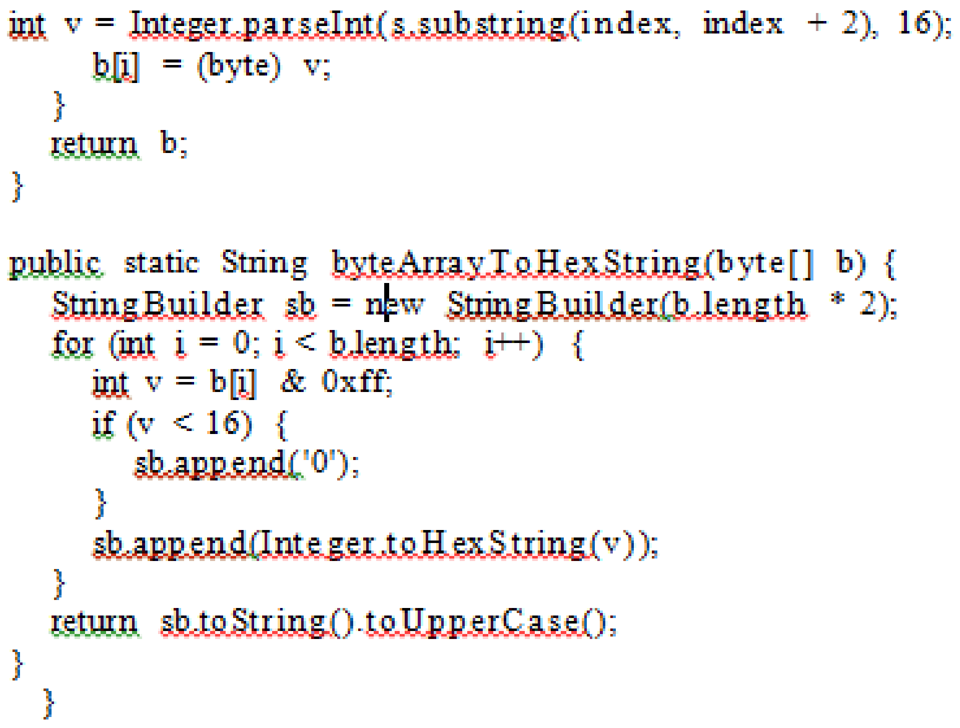
Figure E.8:

KeyManagementModule/RSA.java



```
}
 return "";
}

  public static String displayCharValues(String s) {
   StringBuilder sb = new StringBuilder();
   for (char c : s.toCharArray()) {
      sb.append((int) c).append(",");
   }
   return sb.toString();
}
  public static String encrypt(final String plaintext,String KEY) throws
GeneralSecurityException {
   SecretKeySpec sks = new SecretKeySpec(hexStringToByteArray(KEY), "AES");
   Cipher cipher = Cipher.getInstance("AES");
   cipher.init(Cipher.ENCRYPT_MODE, sks, cipher.getParameters());
   byte[] encrypted = cipher.doFinal(plaintext.getBytes());
   return byteArrayToHexString(encrypted);
}

public static byte[] hexStringToByteArray(String s) {
   byte[] b = new byte[s.length() / 2];
   for (int i = 0; i < b.length; i++) {
      int index = i * 2;
```

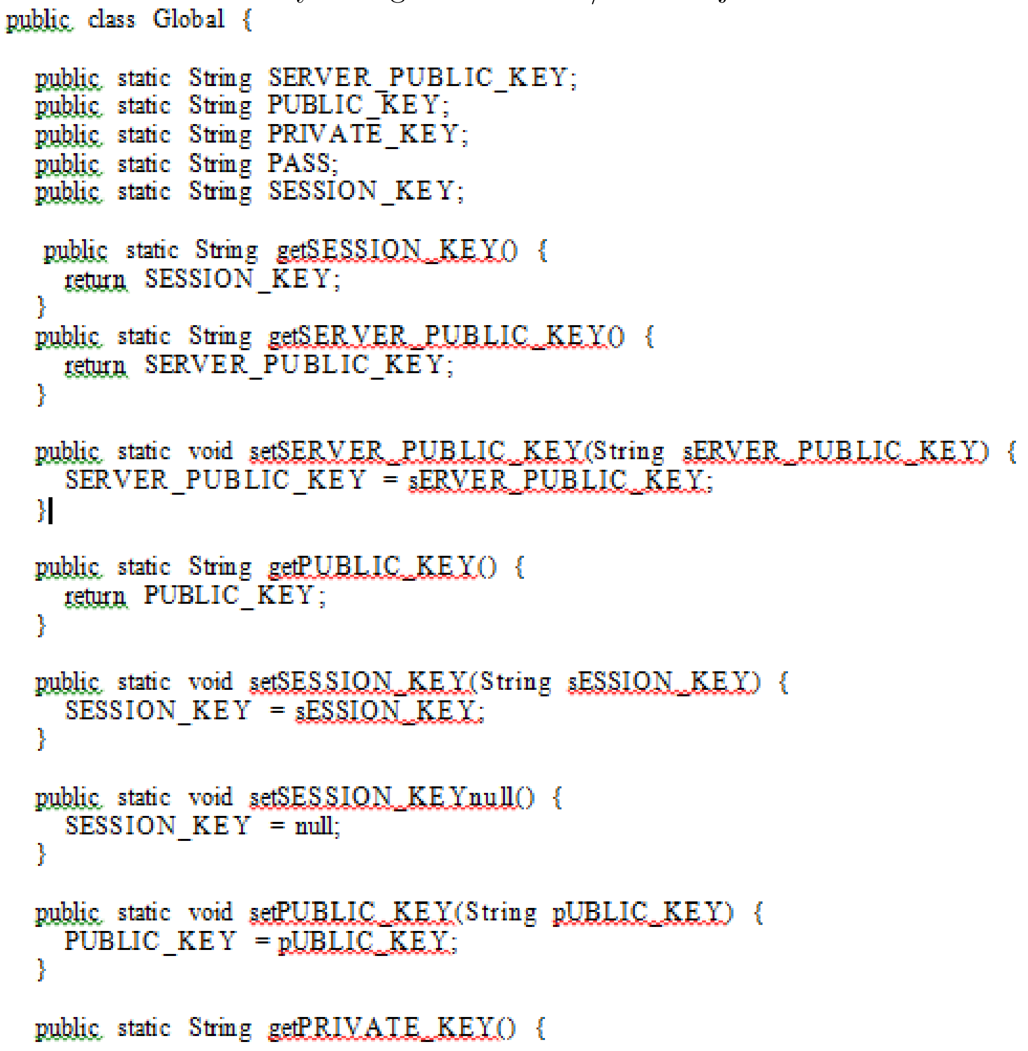
Figure E.9:



```
int v = Integer.parseInt(s.substring(index, index + 2), 16);
        b[i] = (byte) v;
    }
    return b;
}

public static String byteArrayToHexString(byte[] b) {
    StringBuilder sb = new StringBuilder(b.length * 2);
    for (int i = 0; i < b.length; i++) {
        int v = b[i] & 0xff;
        if (v < 16) {
            sb.append('0');
        }
        sb.append(Integer.toHexString(v));
    }
    return sb.toString().toUpperCase();
}
   }
```

Figure E.10:

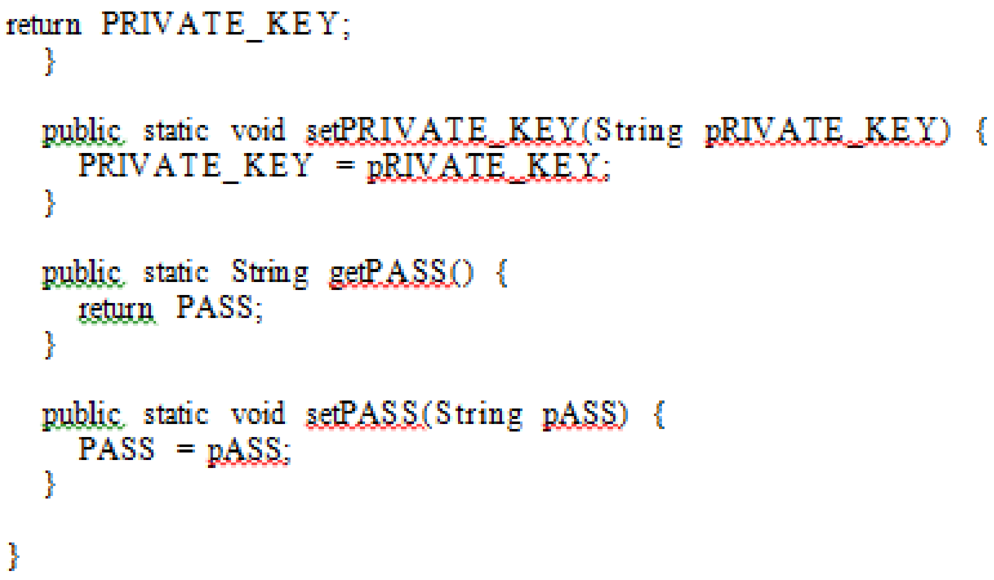
KeyManagementModule/AESnet.java



```
public class Global {

  public static String SERVER_PUBLIC_KEY;
  public static String PUBLIC_KEY;
  public static String PRIVATE_KEY;
  public static String PASS;
  public static String SESSION_KEY;

   public static String getSESSION_KEY() {
    return SESSION_KEY;
  }
  public static String getSERVER_PUBLIC_KEY() {
    return SERVER_PUBLIC_KEY;
  }

  public static void setSERVER_PUBLIC_KEY(String sERVER_PUBLIC_KEY) {
    SERVER_PUBLIC_KEY = sERVER_PUBLIC_KEY;
  }

  public static String getPUBLIC_KEY() {
    return PUBLIC_KEY;
  }

  public static void setSESSION_KEY(String sESSION_KEY) {
    SESSION_KEY = sESSION_KEY;
  }

  public static void setSESSION_KEYnull() {
    SESSION_KEY = null;
  }

  public static void setPUBLIC_KEY(String pUBLIC_KEY) {
    PUBLIC_KEY = pUBLIC_KEY;
  }

  public static String getPRIVATE_KEY() {
```

Figure E.11:

HelperModule/Global.java



```
return PRIVATE_KEY;
    }

    public static void setPRIVATE_KEY(String pRIVATE_KEY) {
        PRIVATE_KEY = pRIVATE_KEY;
    }

    public static String getPASS() {
        return PASS;
    }

    public static void setPASS(String pASS) {
        PASS = pASS;
    }

}
```

Figure E.12:

Bibliography

MulVAL: A logic-based, data-driven enterprise security analyzer [Online]. http://people.cis.ksu.edu/ xou/argus/software/ mulval/readme.html. Accessed: 2014-09-1.

TANAT - Threat And Attack Tree Modeling plus Simulation [Online]. http://www13.informatik.tu-muenchen.de:8080/tanat/. Accessed: 2014-09-06.

The STRIDE Threat Model, Microsoft [Online]. http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx. Accessed: 2014-09-20.

WinGraphviz news, [Online]. http://wingraphviz.sourceforge.net/wingraphviz/. Accessed: 2014-09-10.

(2005). Assertions and protocols for the oasis security assertion markup language (saml) v2.0.

Amir, Y., Coan, B., Kirsch, J., and Lane, J. (2007). Customizable fault tolerance for wide-area replication. In *In Proceedings of the 26th IEEE Symposium on Reliable Distributed Systems*.

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., and Xu, M. (2007). Web services agreement specification (ws-agreement). In *Open Grid Forum*, volume 128.

Bertino, E., Martino, L. D., Paci, F., and Squicciarini, A. C. (2010). Web services threats, vulnerabilities, and countermeasures. In *Security for Web Services and Service-Oriented Architectures*, pages 25–44. Springer.

Bhatti, R., Bertino, E., and Ghafoor, A. (2005). A trust-based context-aware access control model for web-services. volume 18, pages 83–105. Springer.

Biskup, J., Carminati, B., Ferrari, E., Muller, F., and Wortmann, S. (2007). Towards secure execution orders for compositeweb services. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 489–496. IEEE.

Carminati, B., Ferrari, E., and Hung, P. C. (2005). Web service composition: A security perspective. In *Web Information Retrieval and Integration,* *2005. WIRI'05. Proceedings. International Workshop on Challenges in*, pages 248–253. IEEE.

Carminati, B., Ferrari, E., and Hung, P. C. (2006). Security conscious web service composition. In *Web Services, 2006. ICWS'06. International Conference on*, pages 489–496. IEEE.

Castro, M., Rodrigues, R., and Liskov, B. (2003). Base: Using abstraction to improve fault tolerance. volume 21, pages 236–269. ACM.

Charfi, A., Berbner, R., Mezini, M., and Steinmetz, R. (2008). On the management requirements of web service compositions. In *Emerging Web Services Technology, Volume II*, pages 97–109. Springer.

Charfi, A. and Mezini, M. (2005). Using aspects for security engineering of web service compositions. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*, pages 59–66. IEEE.

Coetzee, M. and Eloff, J. H. (2004). Towards web service access control. volume 23, pages 559–570. Elsevier.

Dan, A., Davis, D., Kearney, R., Keller, A., King, R., Kuebler, D., Ludwig, H., Polan, M., Spreitzer, M., and Youssef, A. (2004). Web services on demand: Wsla-driven automated management. volume 43, pages 136–158. IBM.

Elgedawy, I. (2014). Crescent: A reliable framework for durable composite web services management. page bxu019. Br Computer Soc.

feng YAN, D., TIAN, Y., lin HUANG, J., and chun YANG, F. (2013). Privacy-aware {RBAC} model for web services composition. volume 20, Supplement 1, pages 30 – 34.

Ghosh, R. and Naik, V. (2012). Biting off safely more than you can chew: Predictive analytics for resource over-commit in iaas cloud. In *IEEE 5th International Conference on Cloud Computing*.

Goettelmann, E., Fdhila, W., and Godart, C. (2013). Partitioning and cloud deployment of composite web services under security constraints. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on*, pages 193–200. IEEE.

GRANT, E. E. O. I. and INTELLECTUAL, A. L. T. A. (2002). Web services trust language (ws-trust).

Hutter, D. and Volkamer, M. (2006). Information flow control to secure dynamic web service composition. In *Security in Pervasive Computing*, pages 196–210. Springer.

Joshi, J. B., Aref, W. G., Ghafoor, A., and Spafford, E. H. (2001). Security models for web-based applications. volume 44, pages 38–44. ACM.

Karimi, S. and Babamir, S. (2010). Efficient intelligent secure for web service composition. In *International conference on communication engineering*.

Keller, A. and Ludwig, H. (2003). The wsla framework: Specifying and monitoring service level agreements for web services. volume 11, pages 57–81. Springer.

Kordy, P. Adtool. Universit du Luxembourg.

Koshutanski, H. and Massacci, F. (2004). *Interactive Access Control for Web Services*, volume 147 of *IFIP The International Federation for Information Processing*. Springer US.

Lamport, L., Shostak, R., and Pease, M. (1982). The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401.

Lee, A. J., Boyer, J. P., Olson, L. E., and Gunter, C. A. (2006). Defeasible security policy composition for web services. In *Proceedings of the fourth ACM workshop on Formal methods in security*, pages 45–54. ACM.

Liu, A., Li, Q., Huang, L., and Xiao, M. (2010). Facts: A framework for fault-tolerant composition of transactional web services. volume 3, pages 46–59. IEEE.

Madan, B. B., Goseva-Popstojanova, K., Vaidyanathan, K., and Trivedi, K. S. (2004). A method for modeling and quantifying the security attributes of intrusion tolerant systems. volume 56, pages 167–186.

Mao, Y., Junqueira, F. P., and Marzullo, K. (2008). Mencius: building efficient replicated state machines for wans. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, OSDI'08, pages 369–384.

Merideth, M. G., Iyengar, A., Mikalsen, T., Tai, S., Rouvellou, I., and Narasimhan, P. (2005). Thema: Byzantine-fault-tolerant middleware for web-service applications. In *Reliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium on*, pages 131–140. IEEE.

Onditi, V., Dobson, G., Hutchinson, J., Walkerdine, J., and Sawyer, P. (2008). Specifying and constructing a fault-tolerant composite service.

Pallemulle, S., Thorvaldsson, H., and Goldman, K. (2008). Byzantine fault-tolerant web services for n-tier and service oriented architectures. In *Distributed Computing Systems, 2008. ICDCS '08. The 28th International Conference on*, pages 260–268.

Rissanen, E. (2010). Oasis extensible access control markup language (xacml), version 3.0. oasis committee specification 1, 1150.

Sathiaseelan, J. (2013). Architectural framework for secure composite web services. volume 76. Citeseer.

Satoh, F. and Tokuda, T. (2011). Security policy composition for composite web services. *Services Computing, IEEE Transactions on*, 4(4):314–327.

Schneier, B. (1999). Schneier on Security - Attack Trees,Dr. Dobb's Journal [Online],. https://www.schneier.com/paper-attacktrees-ddj-ft.html. Accessed: 2014-09-10.

She, W., Yen, I.-L., and Thuraisingham, B. (2008). Enhancing security modeling for web services using delegation and pass-on. In *Web Services, 2008. ICWS'08. IEEE International Conference on*, pages 545–552. IEEE.

Souza, A. R., Silva, B. L., Lins, F. A., Damasceno, J. C., Rosa, N. S., Maciel, P. R., Medeiros, R. W., Stephenson, B., Motahari-Nezhad, H. R., Li, J., et al. (2009). Incorporating security requirements into service composition: From modelling to execution. In *Service-Oriented Computing*, pages 373–388. Springer.

T. Hayashi, Lepidum, Y. I. (2017). Mutual authentication protocol for http.

Tanaka, M., Ishida, T., Murakami, Y., and Morimoto, S. (2009). Service supervision: coordinating web services in open environment. In *Web Services, 2009. ICWS 2009. IEEE International Conference on*, pages 238–245. IEEE.

Tziviskou, C. and Di Nitto, E. (2007). Logic-based management of security in web services. In *Services Computing, 2007. SCC 2007. IEEE International Conference on*, pages 228–235. IEEE.

van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. (2003). Owl web ontology language reference. volume 31.

Yu, Q., Liu, X., Bouguettaya, A., and Medjahed, B. (2008). Deploying and managing web services: issues, solutions, and directions. *The VLDB JournalThe International Journal on Very Large Data Bases*, 17(3):537–572.

Zhao, W. (2007a). Bft-ws: A byzantine fault tolerance framework for web services. In *EDOC Conference Workshop, 2007. EDOC'07. Eleventh International IEEE*, pages 89–96. IEEE.

Zhao, W. (2007b). Bft-ws: A byzantine fault tolerance framework for web services. In *In Proceedings of the Middleware for Web Services Workshop.*

Zhao, W. and Zhang, H. (2008). Byzantine fault tolerant coordination for web services business activities. 1:407–414.