分类号 密级

U D C 编号

中南大學

CENTRAL SOUTH UNIVERSITY

硕士学位论文

论 文 题 目 Web 服务环境下单点登录与访问控制研究

学科、专业 计算机软件与理论

研究生姓名 张慧

导师姓名及专业技术职务 李建华 教授

2008 年 5 月

MS THESIS

Research on SSO and Access Control in Web Services Environment

Specialty： Computer Software and Theory

Master Degree Candidate： Hui Zhang

Supervisor： Prof. Jianhua Li

School of Information Science and Engineering

Central South University

Changsha Hunan P.R.China

摘　要

面向服务计算（Service-Oriented Computing, SOC）是一种新型的计算模式，它把服务作为基本的组件来支持快速、低成本和简单的分布式甚至异构环境的应用组合。Web 服务技术是基于 SOC 概念当前最有前景的技术，它具有松耦合、平台无关、异构、跨域、动态变化等特征，可以很好的适用于 SOC 环境。但 Web 服务还有许多安全挑战未解决，比如单点登录和访问控制这两个安全领域非常重要的问题。用户在访问跨域的服务时，由于目标服务对其身份是不可知的，所以无法进行验证。传统的一些单点登录方法也不适用于 Web 服务环境。与单点登录类似，访问控制系统由于无法确定用户的身份，也就无法定义用户的角色与权限，同时业务需求的快速变化也使得传统的访问控制方法在管理的扩展性和控制的粒度上无法适应 Web 服务环境。这些问题使得访问控制系统无法对资源进行合理的保护，防止未授权的用户非法访问资源。目前 Web 服务的安全已成为阻碍其发展的重要因素之一。

本文在分析了 Web 服务安全相关技术与规范的基础上,从三个方面对 Web 服务安全做了研究：提出了一种基于 SAML 的 Web 服务单点登录模型，使得服务可以基于 SAML 断言来对跨域未知用户进行验证,给出了模型的两种实现模式，对组合服务的单点登录做了研究，并对模型的安全性进行了分析；提出了一种具有协商机制的基于属性的 Web 服务访问控制模型 Nego-ABAC，它基于用户、环境、资源的属性来进行访问控制的评估，弥补了基于身份、角色的访问控制的缺陷，解决了对跨域未知用户的访问控制问题，引入协商机制，使得访问控制具有更强的自治性；提出了一种基于信任级别和协商机制的 Web 服务环境下用户敏感属性保护模型，在基于属性的访问控制系统中，由于用户提供的是属性，而这些属性中有的是敏感的，所以必须要加于保护，不能泄漏给任意服务提供者方，所以文中提出通过信任级别的比较和服务提供者的身份属性的验证协商过程，来披露敏感属性。最后基于上述三个方面，设计了一个可扩展的 Web 服务安全系统原型 EWSSSystem。

综上所述，本文的工作针对目前 Web 服务安全中几个关键问题提出了有效地解决方案，对于推进 Web 服务安全技术的发展具有一

定的理论价值和应用价值。

关键词 面向服务计算，Web 服务，服务安全，单点登录，访问控制，敏感属性保护

ABSTRACT

Service-Oriented Computing (SOC) is a new computing paradigm that utilizes services as the basic constructs to support the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous environments. Web Services are the current most promising technology based on the concept of SOC. It has characters of loose-couple, platform-independent, heterogeneous, cross-domain, dynamic changes, etc, is suitable for SOC environment. However, Web Services still have many security challenges, such as Single Sign on (SSO) and access control which are the most important domains of security. When users access cross-domain services, it is stranger to target services. So it can't be authenticated. Traditional SSO solutions are not fit for Web Services environment. Similarly to SSO, because of users' identities not known, so it is very hard for access control system to define roles and permissions. At the same time, rapid business requirement changes make traditional access control system not suitable for Web Services environment in administrative scalability and control granularity. Access control systems can't protect resources reasonably to unauthorized users. Currently, security has become one of the most important problems that impeded the development of Web Services.

Based on analysis of some techniques and standards of Web Services security, this paper does research on Web Services in three fields. Firstly, present a SAML-based Web Services SSO model. Services can authenticate users based SAML assertions. Two implementation patterns of model and composite services SSO method are given. Then security of the model is analyzed. Secondly, present an attribute-based access control model for Web Services holding Negotiation Mechanisms (Nego-ABAC). It improves identity-based and role-based access control and solves access control to unknown users. Using negotiation mechanisms, Nego-ABAC becomes more autonomy and adaptive. Thirdly, present user sensitive attributes protect model based trust level and negotiation mechanism in Web Services enviroment. In attribute-based access control

(ABAC), users provide attributes to be evaluated.But some of user attributes is sensitive, need to be protected, and can’t be sent to every service provider. So it need compare trust level, negotiate and authenticate service provider’s identity attributes to disclose sensitive attributes. At last, design an extensible Web Services security system (EWSSSystem) based above three fields.

In summary, this paper present effective solutions solving several key issues in Web Services security. We believe that our contributions make a nice groundwork for future research and engineering on Web services security both in theory and practice.

KEY WORDS Service-Oriented Computing, Web Services, services security, SSO, access control, sensitive attributes protect

目 录

第一章 绪论

1.1 引言

1.1.1 问题的提出

Web 服务是一种支持机器之间通过网络来互操作的软件系统[1]。它建立在 XML 规范、SOAP 协议、WSDL 规范和 UDDI 规范的基础之上，提供了一种使用 HTTP、SMTP 或 FTP 等与 Internet 相兼容的协议来访问业务或者应用程序逻辑的方式。Web 服务基于面向服务计算（Service-Oriented Computing,SOC）[2][3]概念，具有松耦合、平台无关、异构、跨域、动态变化等特点，是面向服务架构（Service-Oriented Architecture,SOA）[4]当前最有发展前景的实现技术。

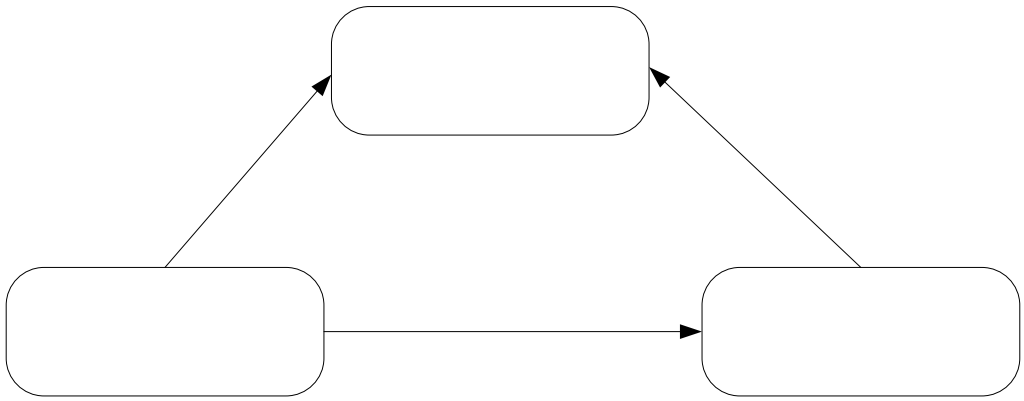

图 1-1 基于 Web 服务的面向服务体系结构

如图 1-1 所示, 基于 Web 服务的面向服务体系结构[5]包括三个参与者：服务请求者、服务注册中心及服务提供者。服务提供者向服务注册中心发布服务， 服务用 WSDL 来描述。服务请求者往 UDDI 服务注册中心查询服务。服务请求者使用 SOAP 协议来调用服务。

随着 Web 服务的发展，越来越多的应用集成将会采用 Web 服务技术[6]。安全是一个应用系统必须要考虑的问题，所以 Web 服务的安全越来越受到人们的关注。目前，对 Web 服务的安全已有一定的研究，比如 XML Signature、XML Encryption、WS-Security，这三个规范结合保证了 SOAP 消息的安全性。但在单点登录和访问控制方面还存在很多不足。

传统的单点登录方法都是基于单域的，各系统的用户信息相同，而 Web 服务环境下的跨域单点登录却有着用户不可知的特点，所以传统的方法无法适用于

服务注

查找服务

Web 服务环境。

与单点登录类似，访问控制系统由于无法确定用户的身份，也就无法定义用户的角色与权限，同时业务需求的快速变化也使得传统的访问控制方法在管理的扩展性和控制的粒度上无法适应 Web 服务环境。这些问题使得 Web 服务环境下的访问控制系统无法对资源进行合理的保护，防止未授权的用户非法访问资源。

1.1.2 本文研究的背景

自从八十年代互联网出现以来，人类社会的信息资源正在以前所未有的程度和方式在全球内互联和互通，互联网已经成为人们生活的一部分，并给人们的工作、生活、学习等各个方面带来巨大影响。

随着互联网技术的迅速发展，基于互联网的应用模式也在不断演变。越来越多的企业和政府部门依赖互联网来发布信息与提供服务，并构建跨企业的虚拟组织或虚拟企业以实现大规模资源共享。但由于互联网环境是一个异构、动态、松耦合的分布式计算环境，因此如何安全、有效、简便地实现基于互联网的资源共享、业务协作便成为了互联网发展道路中的必须要解决的问题。

在传统的互联网应用中，人们习惯使用 HTML 来描述信息资源和服务，但由于 HTML 可扩展性较差，不能准确地刻画数据内容，不利于大规模商业活动的自动化操作，也不利于有效地利用蕴藏在互联网中浩如烟海的资源，从而导致基于互联网的应用在进一步深入发展时受到很大的限制。随着 JAVA 和 XML 技术的出现，解决了跨平台和信息互操作的问题，但由于传统的分布式技术（如 CORBA[7]、DCOM[8]等）都维护着相互独立的体系结构和协议以及客户端和服务器端的紧耦合性，使得它们不能适应动态发布服务的需求，也不能很好地实现在动态、松耦合环境下的业务协作。因此，为了满足日益增长的业务需求，人们提出了 Web 服务概念来解决新一代互联网软件所面临的问题。

Web 服务是一种崭新的分布式计算模式,基于一系列开放的标准技术,如 SOAP、UDDI 和 WSDL。其松散耦合、语言中立、平台无关性、开放性将使它成为下一代电子商务的框架。但同时因为它的松散耦合及跨域的特点，使得传统的安全模型（如单点登录、访问控制）无法适用于 Web 服务环境。目前，Web 服务的安全已经成为阻碍 Web 服务广泛使用的重要问题之一[9][10]。

1.1.3 研究的目的与意义

虽然 Web 服务技术有着广泛的应用前景，但其安全性问题如果不能得到根本性解决，就会制约其发展。尤其是涉及到企业应用的时候，一个不得不关注的

问题就是安全。而 Web 服务的诸多优点反而使得安全问题变得比以往任何时候都更具挑战性。

单点登录和访问控制是系统安全的重要组成部分，在传统的应用中已经得到了较好的解决。但由于 Web 服务的松耦合、跨域等特点使得传统的方法无法适用于 Web 服务环境。所以研究一种适合于 Web 服务环境下的单点登录和访问控制模型，有着很大的意义。

1.2 国内外研究现状

1.2.1 单点登录的研究现状

传统的单点登录技术[11]主要是针对单域环境下的，一般是用统一身份认证技术来实现的，如 LDAP，所有的系统都到一个集中的地方去认证。但对于跨域的系统，由于各系统用户是不一致的，且互相是不可知的。所以传统的单点登录技术在跨域环境下是不可行的。

目前，针对跨域环境下的单点登录解决方案主要有：微软的.NET Passport[12]、Sun Microsystems 等建立的自由联盟计划（liberty alliance project）[13]、Microsoft 和 IBM 联合开发的 Web 服务联邦语言（WS-Federation）[14]以及结构化信息标准促进组织（OASIS）的安全服务委员会（SSTC）提出的安全断言标记语言（Security Assertion Markup Language,SAML）[15]。.NET Passport 技术是通过其 Passport 来实现单点登录的，只要用户通过微软的 Passport 服务器的验证，就可以访问所有与 Passport 服务器合作的站点。但由于微软在 Passport 验证技术方面不公开，使得在安全性方面有一定的隐患。目前，Passport 还不支持 Web 服务。自由联盟计划和 Web 服务联邦语言都是通过建立联盟身份，来访问联盟中的其它系统的。但由于 Web 服务是松耦合的，所以建立联盟身份并不是每个 Web 服务场景所必须的。

SAML 主要用来在不同信任域之间交换安全信息，为认证和授权服务提供了标准的描述，基于 XML 具有跨平台性，提供了强大的断言（Assertion）[16]机制，使得跨域的系统可以通过断言来进行验证，适用于 Web 服务的松耦合环境。

目前对基于 SAML 的 Web 服务单点登录模型学术界和工业界都有了一定的研究。SAML 规范中的 Bindings[17]部分定义了 SAML 如何与 SOAP 协议进行绑定,为 SAML 与 Web 服务的结合提供了标准。文献[18]提出了一种基于 Systinet 公司的 WASP Card 产品来构建 Web 服务单点登录应用，虽然这个产品可以对 SAML 请求进行响应并发布断言等，但这种方法依赖于特定的产品,通用性不强。文献[19]提出了一种基于 SAML 的 Web 服务单点登录设计方案，可以解决单个

服务时的单点登录，但没有提供组合服务时的单点登录方法。文献[20][21]对基于 SAML 的单点登录系统进行了分析，但没有讨论在 Web 服务环境下如何实现单点登录。文献[22]提出一种基于 SRP 和 SAML 的 Web 服务验证控制系统，解决了 Web 服务客户端用户的身份验证问题。但 SRP 协议的握手次数太多，且身份验证方的功能是以 Web 服务的方式发布的，所以会存在大量的 Web 服务调用，虽然安全性较好，但严重影响了性能。

1.2.2 访问控制的研究现状

传统的访问控制模型主要有两类：自主访问控制（Discretionary Access Control, DAC）和强制访问控制（Mandatory Access Control, MAC）。DAC 模型是一种基于身份的访问控制，每个资源都有一个访问控制列表，里面记录可以访问该资源的用户信息，虽然可以保护资源，但很难查找一个用户具有哪些权限，必须要穷举所有访问列表，当用户数和资源数很大时，系统很难管理。在 MAC 模型中，每个主体和资源都被预定义一个安全级别，主体只能读那些安全级别比其低或相等的资源，往那些安全级别比其高或相等的资源中写。这种模型虽然可以保证信息的一致性，但不够灵活，不适合 Web 服务的动态性环境。

随着资源和主体数的增加，系统的授权管理工作变的更加复杂，上述两种模型都不能很好的工作。为简化授权管理工作，提出了基于角色的访问控制（Role Based Access Control, RBAC）[23][24][25][26][27]。RBAC 通过用户与角色、角色与权限的映射，简化了授权任务。RBAC 可以较好的满足传统的应用，但并不能很好的解决 Web 服务中的访问控制问题。因为传统的应用中，用户都是确定的，而 Web 服务的用户是不可知的，通常服务的请求者和提供者都来自不同的域，所以必须要进行角色的映射，增加了管理负担，而且这个方法只能用在静态的服务中，当在动态的服务流程中，服务都是不确定的，由服务执行引擎去动态选取，所以进行角色的映射是不可能的。其次，当业务场景非常复杂时，RBAC 只能制定更多的角色，导致了角色扩散问题，大大加重系统的负担。

基于上述情况，国内外已进行了不少相关研究，提出了基于属性的访问控制（Attributed Based Access control, ABAC）模型，见文献[28][29][30][31]。在 ABAC 中，当业务变化时，不需要重新定义角色，只要简单的根据属性来修改策略即可。文献[28]提出了一种带有协商机制的 ABAC 模型，可以协商服务的参数，但对用户的属性并为涉及。文献[29]也提出了一种 ABAC 模型，并与 RBAC 模型做了详细的比较，证明了 ABAC 模型的先进性，但对协商机制并未涉及，自治性较差。文献[30][31]提出的 WS-ABAC 模型，结合了自动信任协商技术，主要侧重点在于建立信任，保护敏感属性，并未讨论服务请求授权的自治性，比如用户通

过动态调整参数和属性来适应服务提供者的访问策略。文献[32][33][34][35][36]提出了基于 XACML 的访问控制模型，XACML 规范支持基于属性的访问控制，但它并不支持协商机制，自治性较差。

1.2.3 ABAC 中敏感属性保护的研究现状

目前，在 Web 服务环境下，基于属性的访问控制被认为是最适合的访问控制方法[28-31]。然而在基于属性的访问控制中，用户有的属性是敏感的，不能泄漏给服务提供者，但是服务提供者在访问控制时却需要该属性，这时就需要与用户进行协商。在这个过程中需要建立策略来对敏感属性进行保护。

文献[37]中给出了一种敏感属性保护方法，它用信任级别来描述具体的服务及用户属性，当服务的信任级别高于用户属性时，就可以访问该属性。这种方法较好的解决了用户访问静态服务时敏感属性的保护问题。但 SOC 中提出服务可能是动态选取组合的，所以该方法有一定的局限，同时它提出的信任级别是由用户自己指定的，这在实际应用中是不现实的，须由第三方如服务注册中心来指定，且该方法不具有协商功能，自治性较差。

文献[30]提出了基于 TrustBuilder[38]的自动信任协商体系结构来解决敏感属性保护问题的思路，但没有详细介绍这种方法，比如在协商过程中的一些策略，所以在具体的系统设计与实现时有很多问题还有待解决。

文献[39][40][41][42][43][44]都给出了允许资源请求者和访问控制中介者建立互相信任的自动信任协商（Automated Trust Negotiation,简写为 ATN）[45][46]机制，协商双方使用协商器组件来决定是否把敏感属性泄漏给对方，先互相泄漏非敏感的属性，等建立了一定的信任后，再把敏感属性泄漏给对方。但在上述的 ATN 系统设计过程中对属性的拥有情况没有做保护，攻击者可以很轻易的模仿协商的一方获得协商的另一方是否拥有某种属性。

文献[47]提出了一种在基于属性访问控制环境下的自动信任协商机制，它使用了 Ack 策略、TTG 协议以及基于角色信任管理语言（RT），较好的解决了上述问题。其中 Ack 策略是指协商的一方为敏感属性建立的一种访问控制策略，它定义了协商另一方只有满足了 Ack 策略，敏感属性才可以流向它。在满足 Ack 策略之前，协商的一方不会泄漏这个属性及是否拥有这个属性的信息。Ack 策略与访问控制策略不同之处在于：主体可以使用 Ack 策略来保护其并不一定拥有的属性。所以基于它可以对敏感属性进行很好的保护。

文献[39-44][47]虽然都是对跨域环境下的访问控制做研究，但他们考虑的应用场景比较局限，只适应很少的一些服务之间的协商。面向服务计算提出未来的应用都是以服务的形式发布，当服务较多时，Ack 策略将无法适应。因为不是所

有的服务提供者都有用户 Ack 策略中要求的属性值。所以势必要定义许多属性条件，这样 Ack 策略将非常庞大，不易于管理，增加了用户的负担。

1.3 研究内容

Web 服务环境具有松耦合、跨域、分布式、跨平台等特性，所以传统的单点登录和访问控制不适用于 Web 服务环境。所以本文对 Web 服务环境下的单点登录和访问控制做了一定的研究。

1.研究一种基于 SAML 的 Web 服务环境下的单点登录模型

由于 Web 服务具有异构性和松耦合性等特点，而现有的跨域单点登录方案主要针对基于 Web 站点的应用，不能较好的适用于 Web 服务环境。SAML 具有平台中立性及强大的断言机制，非常适合 Web 服务环境下的单点登录。所以本文提出一种基于 SAML 的 Web 服务单点登录模型，介绍了该模型的两种实现模式,并给出了组合服务的单点登录方法，最后分析了模型的安全性。

2.研究一种具有协商机制的基于属性的 Web 服务访问控制模型

Web 服务技术的发展使得基于 Internet 的分布式、异构的应用集成起来非常的方便，但其同样也带来了非常大的安全性挑战。传统的访问控制系统大都基于用户的身份，在管理的扩展性和控制的粒度上都存在一些问题，不能很好的适用于 Web 服务环境。提出一种基于属性的访问控制方法，可以适应快速业务变化及复杂的授权关系。同时引入了协商机制，服务提供者与用户可以就访问请求中的服务参数和用户属性进行协商，使得访问控制具有更强的自治性和自适应能力。

3.研究一种基于信任级别和协商机制的用户敏感属性保护模型

基于属性的访问控制机制比较适合于开放的 Web 服务环境。在 ABAC 中，授权决策是基于请求者的属性，而不仅仅是身份。用户在发送服务请求时，会提供自己的属性。这些属性有的是敏感的，不能提供给任意服务。同时服务提供者在像身份提供者请求属性时，请求的可能是用户的敏感属性。这些情况都可能会造成用户敏感属性的泄漏。所以本文研究了一种基于信任级别和协商机制的 Web 服务环境下敏感属性保护模型。

4.设计一个可扩展的 Web 服务安全系统原型

在以上研究的基础上，设计一个可扩展的 Web 服务安全系统原型。基于开源项目 OpenSAML1.1API、WSS4J1.5.3 、SunXACML1.2，设计单点登录、访问控制与协商、用户敏感属性保护等模块，并最终实现该原型。

1.4 本文的组织结构

本文的组织结构如图 1-2 所示。

第一章，绪论：提出了当前 Web 服务环境下的存在的一些安全问题，并对当前的研究现状进行了分析，最后给出了本文的研究内容。

第二章，Web 服务安全相关技术与规范：介绍了 Web 服务安全中的基本规范及协议，XML 签名、XML 加密、WS-Security，以及单点登录及访问控制中用到的 SAML 和 XACML 规范。

第三章，一种基于 SAML 的 Web 服务单点登录模型：设计了一种基于 SAML 的 Web 服务单点登录模型，借助于 SSL 和 PKI 技术，实现了跨域的 Web 服务单点登录。同时给出了模型的两种实现模式，并基于其中一种模式实现了组合服务的单点登录。最后对模型的安全性进行了分析。

第四章，一种具有协商机制的 Web 服务访问控制模型：提出了一种基于属性的具有协商机制的访问控制模型（Nego-ABAC），给出了访问请求和访问控制策略的形式化描述、访问控制规则、协商机制以及访问控制的算法。最后给出了基于 XACML 的 Nego-ABAC 体系结构，并对模型的安全性进行了分析。

第五章，基于信任级别和协商机制的用户敏感属性保护研究：提出了一种基于信任级别和协商机制的 Web 服务环境下用户敏感属性保护模型，并对其进行了形式化描述，制定了一种 TLBAck 策略来保护敏感属性,给出了相关算法。对组合服务下的敏感属性保护做了研究，最后对模型的安全性进行了分析。

第六章，可扩展的 Web 服务安全系统原型设计：设计了单点登录及访问控制原型 EWSSSystem，在开源项目 OpenSAML1.1API、WSS4J1.5.3、SunXACML1.2 的基础上，对各模块进行了设计。

第七章，总结和展望：总结本文工作，并指出进一步的研究方向。

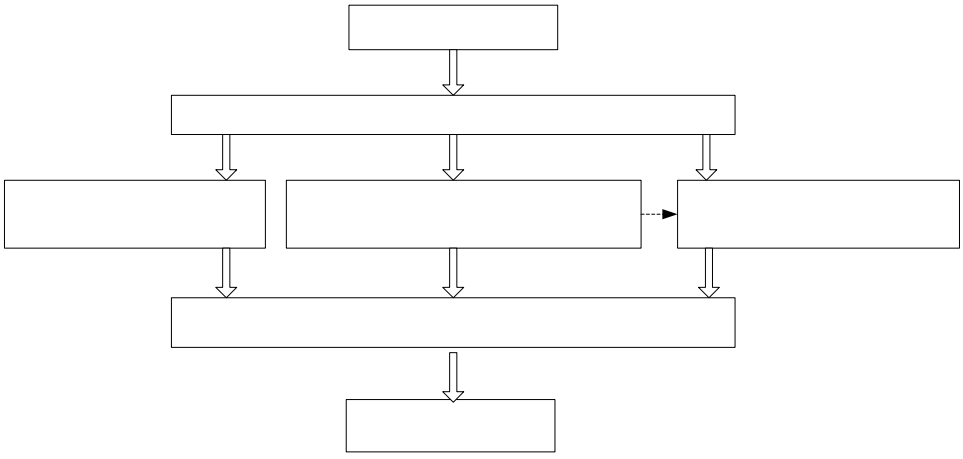

图 1-2 本文的组织结构

第二章 Web 服务安全相关技术与规范

由于 Web 服务环境下的单点登录和访问控制中为了保证安全性均采用了加密和签名。所以本章首先对加密和签名过程中涉及的三个规范 XML 签名、XML 加密、WS-Security 进行了介绍。WS-Security 结合 XML 签名和 XML 加密，将使得 SOAP 消息更加安全，机密性、完整性、验证性、不可抵赖性都得到了保证。然后再分别介绍了单点登录中用到的 SAML 规范及访问控制中要到的 XACML 规范，这两个规范在 Web 服务环境下使用时，它们的底层传输都是基于上述三个规范，所以说这五个规范是紧密结合的，共同保证了 Web 服务环境的安全。

2.1 XML 签名规范

XML 签名（XML Signature）规范[48]是由 W3C 和 IETF 一个合作工作小组（XML Signature WorkingGroup）于 2002 年开发出来的规范。该规范描述了数字签名的 XML 表示及计算、验证 XML 表示的数字签名的过程。

XML Signature 包括以下的功能：

1．XML Signature 可以对任何能够以 URI 形式（uniform resource identifier）定位的资源做签名。既包括与签名同在一个 XML 文件中的元素，也包括其他 XML 文件中的元素，甚至可以是非 XML 形式的资源（比如一个图形文件），只要能被 URI 定位到的资源都可以应用 XML Signature. 这也代表了 XML 签名的对象可以是动态的变化。

2．XML Signature 可以对 XML 文件中的任一元素做签名，也可以对整个文件做签名。

3．XML Signature 既可以用非对称密钥做签名（Digital Signature），也可以用对称密钥做签名（HMAC）。

采用传统的数字签名同样可以保证整个 XML 文档或消息的不可否认性、身份认证和数据完整性。但这种情况却不符合 Web 服务的应用环境。因为如果对整个文档或消息进行数字签名，则签名后不能再对其进行修改（添加或删除元素）。如果能实现对 XML 文档或消息的细粒度控制，则更能适应 Web 服务的灵活环境。引入 XML 签名后，可在签名后的文档内继续添加签名，同时也能实现对文档内容的可选择签名，根据需要选择不同的元素或元素的属性进行数字签名。在 Web 服务环境中，由不同机构运行的 Web 服务可能会进行必要的联合，以共同完成一项特殊的任务，此时 XML 签名的这种能力就显得格外重要。

2.2 XML 加密规范

XML 加密（XML Encryption）规范[49]是是 W3C 的另一个工作小组（XML Encryption Working Group）于 2002 年开发出来的规范。这个规范描述了对数据的加密过程及加密结果的 XML 表示，这些数据既可以是一个完整的 XML 文档或一个 XML 文档中的指定元素，也可以是从外部引用的任意非 XML 格式数据。加密结果随后被表示为一个 XML 加密元素，这个 XML 加密元素既可以直接含有加密的数据，也可以间接地从外部引用加密的数据。

利用 XML Signature, 消息的完整性（Integrity）得到了保证。虽然签名可以保证消息在传送的途中没有被篡改，但是并不能避免它被偷取。如果消息没有经过加密，那么某个敏感的信息就会被泄漏。XML Encryption 是结合了 XML 技术和传统加密技术而产生的，主要提供了以下的功能：

1. 加密整个 XML 文件。
2. 加密 XML 文件中的某个元素。
3. 加密 XML 文件中某个元素的内容。
4. 加密非 XML 格式的资源，如一张 JPEG 图片。
5. 加密已经过加密的内容。

2.3 WS-Security 规范

WS-Security 规范[50]是最初由 IBM、Microsoft 和 Verisign 公司共同提出的一个协议规范。该规范主要提供了三种机制：安全性令牌传送，消息完整性和消息机密性。安全性令牌包括普通的用户名/密码令牌、X.509 证书令牌、SAML 令牌。消息的完整性和机密性是通过引入 XML 数字签名和 XML 加密协议实现的。这些机制可以单独使用，也可以组合使用，以实现不同强度的安全性。WS-Security 当前的最新版本是 Web Services Security v1.1，由 OASIS 于 2006 年 2 月发布。

WS-Security 规范实际上是对 SOAP 协议的扩展规范，本身并没有提出新的加密算法或安全模型,它只是提供了一个框架,用户可自由地将 Web 服务协议应用层协议与各种加密技术安全模型结合起来,以实现 Web 服务环境下消息的完整性，保密性和消息的认证。具体而言,它在 SOAP 中引入现有的 XML 数字签名和 XML 加密协议。根据这些标准,它定义了一系列 SOAP 消息报头以包含数字签名加密信息和安全令牌等安全信息。

2.4 SAML 规范

2.4.1 SAML 概述

2002 年 12 月，结构化信息标准促进组织（OASIS）批准 SAML V1.0[51]成为其正式标准,并于 2005 年发布了 SAML V2.0[52]。SAML 是 OASIS 下属的安全服务技术委员会（SSTC）所制定的安全服务标准，主要用来在相互信任的合作者之间交换安全信息。它基于 XML 的框架，集成了 XML Signature, XML Encryption 和 SOAP 等其它规范。

SAML 可以解决 Web 服务安全体系中的身份认证多次使用的问题，能为用户提供跨越异种网络和平台的单点登录的认证和授权，尤其是在 Web 服务的系统和流程合作的基础上的，允许多个系统共同分享安全问题和身份认证方面的信息，即身份认证的信息可以在多个服务中传递，从而免除了多次认证的麻烦，进而提高了安全化网络服务的性能。SAML 并不定义任何新的认证和授权机制或方法，只定义用于不同域的服务间安全信息传输的文档结构，确切地说，它是一种语言，进行单一的 XML 描述，允许不同安全系统产生的信息进行相互交换。

2.4.2 SAML 组成

SAML 规范由断言、协议、绑定和配置文件这四个部分组成。

1. 断言（Assertion）

断言是 SAML 的基本数据对象。它是对主体的身份、属性、权限信息的 XML 描述。它是由专门的 SAML 权威（SAML Authority）生成和发布的。SAML 权威一般称作断言方或身份提供者，而接受断言的一般称作信任方。信任方根据主体的断言来进行验证和授权，以决定是否允许主体来访问服务，所以也称作服务的提供者。

所有断言都由下列四类要素组成：

1) 基本信息（Basic information）：与断言构造相关的信息，SAML 规范的版本号，断言的唯一标识，断言发行者的唯一标识，断言的发行时间及断言的有效期。

2) 断言陈述（Assertion Statement）：一个或多个由断言发行者生成的陈述（Statement）。包括：主体陈述（Subject Statement）、认证陈述（Authentication Statement）、属性陈述（Attribute Statement）、授权决议陈述（Authorization Decision Statement）。一个断言标记中可以包含多个陈述。

3) 条件（Conditions）：断言的状态可能受某些条件的限制。如断言的有效

性可能依赖于来自某种确认服务的信息。

4) 建议（Advice）：断言中可以包含一些额外的，由发行者提供的用于认证或授权的辅助信息。

2. 协议

SAML 定义了许多请求/响应协议，这些协议规定了两点间共享 SAML 数据（断言）所需交换的消息种类和格式，而两点间的消息传输通过与具体传输协议的绑定来实现，由于可以与多种标准的传输协议或 XML 消息交互框架绑定（如基于 HTTP 的 SOAP），SAML 具有良好的开放性。将来，SAML 请求和响应格式将有可能绑定到其它通信和传输协议。

SAML 给出两种消息格式：请求消息（Request）和响应消息（Response）。请求消息中可以包含四种类型的查询（Query），分别是主体查询（SubjectQuery）、认证查询（Authentication Query）、属性查询（Attribute Query）和授权决策查询（Authorization Decision Query），分别对应于不同断言的查询。对所有查询，SAML 采取统一格式的相应消息封装对请求的响应结果。

3. 绑定

绑定定义了如何通过标准的传输协议和消息收发协议交换 SAML 消息。例如，SAML SOAP 绑定说明了 SAML 请求和响应消息交换被转换为 SOAP 消息交换。图 2-1 对此作出了解释，并说明了 SAML 协议在 SOAP 消息中的位置。

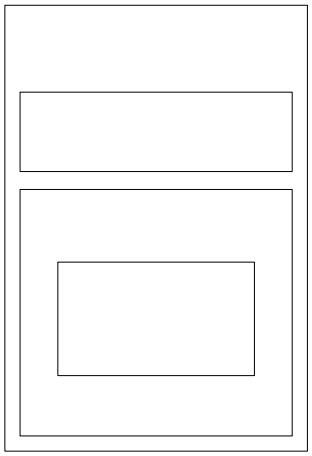

图 2-1 SAML 协议与 SOAP 的绑定

SAML 可以与多种通信和传输协议捆绑在一起，它可以与 HTTP 上的简单对象访问协议（SOAP）相连接。SAML SOAP 绑定规定了如何使用 SOAP 消息来传输 SAML 请求和响应，当使用 SOAP 绑定时，SAML 信息（即 SAML 断言）就是一个 SOAP 消息，存放在 SOAP 消息体中。当 SOAP 消息用于从 SAML 机构来回地发送 SAML 断言的请求或者响应的时候，SOAP 消息体中只能有 SAML 信息，而不能有其他信息，该规范同时规定 SOAP 体中的请求或者响应不能多于

一个。绑定的目的不是保证 SOAP 消息的安全，因此对于断言这类的敏感信息，应该在 SAML 机构和请求者之间执行身份认证，同时还需要保证 SOAP 消息的机密性和完整性。

4. 概要文件（profile）

SAML 概要文件定义一些基于 SAML 的特定应用的约束和扩展，主要是为了增加扩展性。例如，Web Browser SSO 概要文件指出 SAML 验证断言在身份提供者和服务提供者之间传递。Web SSO 概要文件详细描述了 SAML 请求/响应协议和其它协议（如 HTTP POST、HTTP Artifact、SOAP bindings）的结合。

另一种 SAML 概要文件是属性概要文件。SAML 定义了一系列的属性概要文件，来说明 SAML 属性断言中的互操作规则。例如：X.500/LDAP 概要文件描述了怎样在 SAML 属性断言中加载 X.500/LDAP 属性。

2.4.3 SAML 优点

SAML 的优点主要包括：

1. 平台中立性

SAML 提出的安全框架，不依赖于具体的平台及特定的厂商实现。它使得安全更加独立于应用逻辑，这点正是面向服务的体系结构所需要的。

2. 提高了终端用户体验

用户只需在身份提供者方验证，就可以直接访问服务提供方提供的服务，无需再次提供验证信息，安全信息即得到了保护，同时用户的体验也提高了。

3. 减少了服务提供者的管理成本

使用 SAML 断言来进行验证，使得服务提供者方无需维护用户的身份信息，降低了服务提供者的管理成本。

2.5 XACML 规范

2.5.1 XACML 概述

2003 年 2 月，OASIS 批准 XACML（extensible Access Control Markup Language）1.0 版本[53]成为 OASIS 标准。2003 年 8 月，OASIS XACML 技术委员会（Technical Committee，TC）批准 XACML1.1 版本[54]成为 OASIS 委员会规范（OASIS Committee Specification）。目前，XACML2.0[55]版草案 2（Committee draft 02）已于 2004 年 9 月 30 日发布。2005 年 12 月 XACML2.0 和相关的文档被批准为 OASIS 标准。

简单的说，XACML 是一种对访问控制策略和访问控制请求/响应产生过程加以描述的语言，XACML 不仅提供了一系列逻辑算法对整个授权过程进行控制，而且提供了支持定义新功能、数据结构、合成逻辑算法等的标准可扩展点。可以根据主体、资源、环境的属性以及所采取的行为进行控制是允许还是拒绝。实际上，返回的结果有四种:允许、拒绝、无法决定（Indeterminate）和不适用（NotApplicable）。

XACML 使用 XML 作为其描述语言是很自然的事，XML 元语言的特性、可扩展的语法和语义特性以及广泛的支持性使其能够对访问控制思想实现最好的支持。

2.5.2 XACML 组成

XACML 规范主要由访问策略语言、数据流模型、上下文处理器等部分组成。

1. 访问策略语言

访问策略语言中最主要的三个部件：规则（Rule）、策略（Policy）、策略集（Policy Set）。规则中定义了对访问请求做出评估的条件，评估的结果有上述四种类型（允许、拒绝、无法决定和不适用）。一个策略中可以有 0-n 个规则，同样一个策略集中也是有 0-n 个策略。

2. 数据流模型

数据流模型主要讲了怎样对访问请求进行接收、评估，并生成响应。XACML 数据流模型由多个组件组成，主要包括策略执行点（PEP）、策略决策点（PDP）、策略管理点（PAP）、策略信息点（PIP）等。PEP 是接收访问请求，PDP 对访问请求进行评估，PAP 对策略进行管理，PIP 获取环境属性和资源属性。

3. 上下文处理器

上下文处理器是把特定域的评估请求，转换成 PDP 可以识别的请求消息，同时把 PDP 生成的评估结果转换成特定域可以接收的格式。同时在评估过程中，它会从 PIP 中取资源和环境属性。上下文处理器也是 XACML 数据流模型中重要的一个组成部分。

2.5.3 XACML 优点

与以往的策略语言相比，XACML 具有许多独特的优点：

1. XACML 是基于 XML 标准的，具有能够同时被人和计算机识别的特点，并且提供了一个标准化的访问控制决策模型，为不同的安全域提供了互操作性;

2. 由于它充分考虑了主体、资源和环境的属性，其资源访问策略是基于主

体、资源和环境的属性，而不仅仅是基于请求者的身份，所以可以提供比基于身份的、简单地拒绝或授权访问更细粒度的控制访问机制;

3. XACML 不但处理授权请求，而且还定义了一种机制，用来创建进行授权决策所需的规则、策略和策略集的完整基础设施。特别地，它可利用规则组合算法将来自不同授权实体的多个策略结合成唯一可用的策略集，为位于分布式系统中的资源进行访问控制决策;而且单个的访问策略可被用于分布在不同管理域的异构资源。

2.6 本章小结

本章分别介绍了五个规范：XML 签名、XML 加密、WS-Security、SAML 及 XACML。这五个规范是互相结合的，WS-Security 使用 XML 签名和 XML 加密来保证 SOAP 消息的机密性，完整性，不可抵赖性，同时使用 SAML 令牌做为验证的一种方式。XACML 作为访问控制语言，防止了未授权用户访问资源。这五个规范共同保证 Web 服务环境的安全。

第三章 一种基于 SAML 的 Web 服务单点登录模型

3.1 引言

随着 Web 服务技术规范和标准的不断发展，Web 服务已经逐渐从理论研究转向实际应用。很多跨域的系统集成都是使用 Web 服务来实现的。但用户在访问这些 Web 服务时，需要多次输入口令或者其它身份凭证。这有两个缺陷:1）每个服务都必须要维护用户的信息，加大了系统的管理任务，同时造成用户的不便； 2）用户的信息可能是敏感的，不能被一些服务所共享。所以，在 Web 服务环境下使用单点登录技术变得非常的重要。

本章设计了一种基于 SAML 的 Web 服务单点登录模型，借助于 SSL 和 PKI 技术，实现了跨域的 Web 服务单点登录。同时给出了模型的两种实现模式，并基于其中一种模式实现了组合服务的单点登录。最后对模型的安全性进行了分析。该模型使得跨域的 Web 服务单点登录比较容易实现,且安全性较好，可适用于各种跨域的基于 Web 服务的应用场景中。

文献[18]提出的借助 Systinet 公司的 WASP Card 产品来发布断言，具有一定的厂商局限性，而本文提出了一个身份提供者方的通用模块结构，具有较强的灵活性。文献[19]提出的单点登录方法只研究了单个服务的情况，对于组合服务中的单点登录它没有做研究，而本文不但给出了单个服务单点登录的模式，而且给出了组合服务中，如何进行单点登录。文献[22]提出的基于 SRP 和 SAML 的 Web 服务验证控制系统，由于身份提供者方在验证时需要较多的握手次数，虽然安全性较好，但性能较低。而本文模型中的身份提供者可以支持多种用户身份验证方式（比如用户名密码、数字证书等），具有较强的灵活性。

3.2 基于 SAML 的单点登录模型设计

3.2.1 SAML 分析

SAML 是一种完全基于 XML1.0 的描述语言，它继承了 XML 的跨越平台的数据表述特性。它的主要设计目标是为了解决互连网上不同站点的单点登录问题。由于传统的单点登录方案，存在互操作性困难等一些问题，而基于 XML 的 SAML 具有跨平台，语言中立等特征，可以适用于跨域的单点登录。

SAML 的核心思想是基于断言（assertion）来进行身份认证。断言由源站点

生成，里面包含一些断言陈述（assertion statement），比如验证断言陈述、属性断言陈述、授权决议陈述。另外还包括主体 saml:Subject，以及断言的一些条件 saml:Conditions。最后断言生成方对断言进行数字签名。用户使用断言登录目标站点时，目标站点只须验证断言即可确认用户的身份。过程如下：1）进行签名的确认，确定是否是受信任方发布的断言；2）验证断言的条件 saml:Conditions（比如 NotBefore、NotOnOrAfter 以及断言方自定义的一些条件）是否满足；3）主体的确认 SubjectConfirmation，包括有三种方式：holder-of-key、sender-vouches、bearer，用来让目标站点对 subject 进行验证，判断主体是否就是断言的发送者，防止第三者窃取断言后冒充主体来发送断言。其中 holder-of-key 是通过主体拥有的密钥来验证，主体的公钥包含在断言的 ds:KeyInfo 中，主体用自己的私钥来签名一段消息，服务提供者通过验证签名即可判断消息的发送者是否就是断言的主体；sender-vouches 由身份提供者来担保用户的身份，用户通过身份提供者来向服务提供者发送请求，即身份提供者用自己的私钥对请求消息进行签名；bearer 方式是主体可以通过 SubjectConfirmationData 中的一些属性来证明消息的发送者就是断言的主体。SAML 通过上述机制来进行跨域站点的单点登录。

SAML 规范中并没有明确定义支持 Web 服务环境下的单点登录，但它定义了 SAML 消息如何与 SOAP 消息的绑定，使得服务提供者可以通过 SOAP 协议来向身份提供者请求断言。WS-Security 规范也已经开始支持 SAML，它把 SAML Token 放入 SOAP 消息头，并使用 SAML 断言中的主体的私钥（holder-of-key）或身份提供者的私钥来对 SOAP 消息进行签名（sender-vouches），供服务提供者进行身份验证。所以，基于 SAML 来实现 Web 服务环境下的单点登录具有可行性，下文将会提出一个基于 SAML 的 Web 服务单点登录模型。

3.2.2 模型总体结构

SAML 技术规范中定义了两个角色，分别为断言方（Asserting party）和信任方（Relying party）：1）断言方即身份提供者，是一个系统或管理域，它存有用户的信息，对用户进行验证后（如使用传统的用户名密码方式或数字证书方式等），发放断言。断言中包含验证断言陈述和属性断言陈述，证明用户何时以何种方式被验证通过了，以及具有哪些属性。2）信任方也即服务提供者，也是一个系统或管理域，信任身份提供者发放的断言，并依据断言来对用户进行验证。由于身份提供者和服务提供者是基于 PKI 互相信任的，所以用户可以用获得的断言来对服务进行访问。服务提供者提供的服务既可能是单个服务，也可能是多个服务的组合。每个服务需要对断言进行检验，以确定用户的身份，然后通过访问控制系统，来决定该用户是否有权限来访问目标资源。

如图 3-1 所示，模型主要分为四个部分：身份提供者、服务提供者、终端用户、服务调用代理。身份提供者和服务提供者之间的通信是双向安全通道，因为他们各自都拥有数字证书。而用户与身份提供者和服务提供者的通信是单向的安全通道，因为用户可能是没有数字证书。用户通过服务调用代理来和服务提供者交互。服务调用代理动态选取服务，并调用服务，使得用户不直接和服务提供者交互，体现了面向服务架构的松耦合性及服务的透明性。

身份提供者中包括身份验证模块、断言及助诊文件生成模块、数字签名模块、断言响应模块。

1) 身份验证模块主要是对用户提供的身份信息，比如用户名密码或数字证书等进行验证，判断其是否为合法用户。

2) 断言及助诊文件生成模块是用来构造断言或断言助诊（Artifact，对断言的一个引用）[16]。

3) 数字签名模块是用身份提供者的私钥对整个断言或助诊文件进行签名。

4) 断言响应模块是根据服务提供者的助诊文件找到相应的断言文件，并返回给服务提供者。

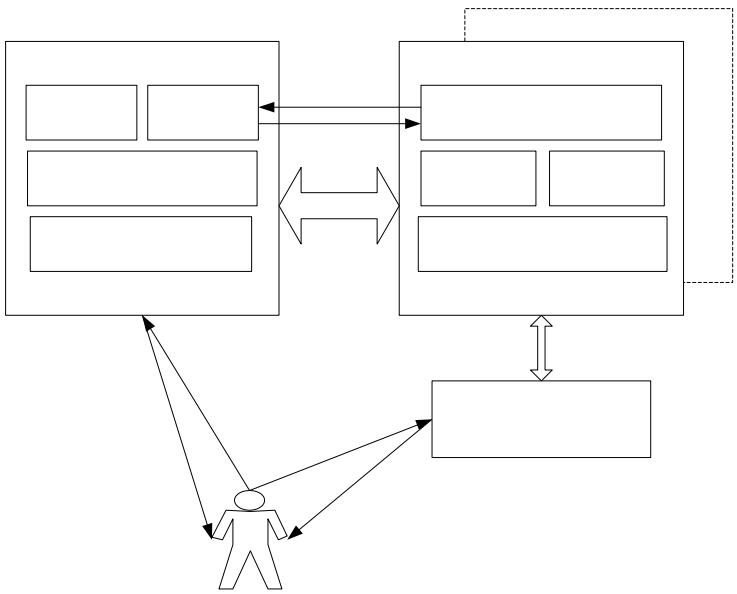

图 3-1 基于 SAML 的 Web 服务单点登录模型

服务提供者中包括断言处理、助诊处理、访问控制、签名确认等模块。

1) 断言处理模块主要是通过对断言的解析来认定用户的身份，并把相关的属性传递给访问控制模块。

2) 助诊处理模块是把客户端用户发送过来的助诊文件传递给身份提供者，并获取对应的断言文件。

身份提供者

助诊文件

身份验证 断言响应

3) 访问控制模块是根据用户的相关属性来判断该用户有无访问资源的权限。

4) 签名确认是用身份提供者的公钥对断言或助诊文件进行确认，判断是否是由身份提供者发布的。

3.2.3 执行机制

整个模型的运转情况如图 3-2 所示：

1）终端用户向身份提供者发出身份验证请求。

2）身份提供者通过身份验证模块对用户进行验证，如果是合法用户，则调用断言构造模块去构造相应的断言；否则拒绝请求。

3）构造断言。

4）身份提供者对断言进行数字签名。

5）返回断言给用户。

6）用户访问目标服务，把请求消息先发给服务调用代理，请求信息中包含了断言信息。

7）服务调用代理来调用具体服务。

8）服务提供者在接收到用户请求时，首先对断言的签名信息进行确认，如果是身份提供者发放的断言，则进行解析，否则拒绝请求。

9）解析该断言，对该用户的身份进行确认。

10)调用访问控制模块来查询该用户有没有权限访问目标资源，并最后给出响应。

上述是直接发放断言情形的执行机制，对于发放助诊文件以及组合服务的情形基本上是类似的，在后面的章节中对这些情况将做详细的论述。

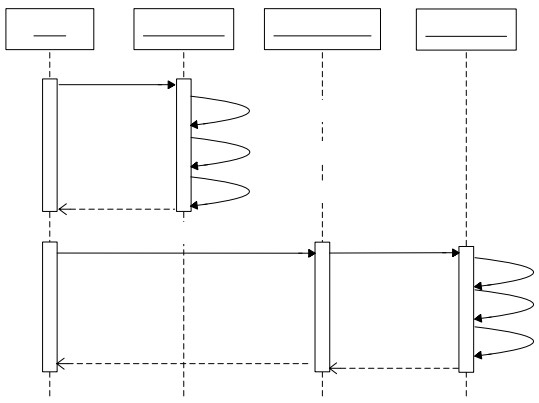

图 3-2 模型的执行机制

3.3 模型的两种实现模式

本节给出了两种 Web 服务单点登录模式 Push 和 Pull，并对这两种模式进行了比较，最后基于 Push 模式给出了组合服务的单点登录方法。

3.3.1 Push 模式

Push 模式是指把断言推给服务提供者，这种模式较为简单，具体的执行步骤如图 3-3 所示：

1) 用户先请求身份提供者，输入自己的身份信息。
2) 身份提供者给用户返回一个 SAML 断言。
3) 用户把包含有断言的服务访问请求发送给服务调用代理。
4) 服务调用代理调用具体的服务。
5) 服务提供者根据获得的 SAML 断言，进行验证，以判断是否是接受请求还是拒绝请求。
6) 把服务调用结果返回给用户。

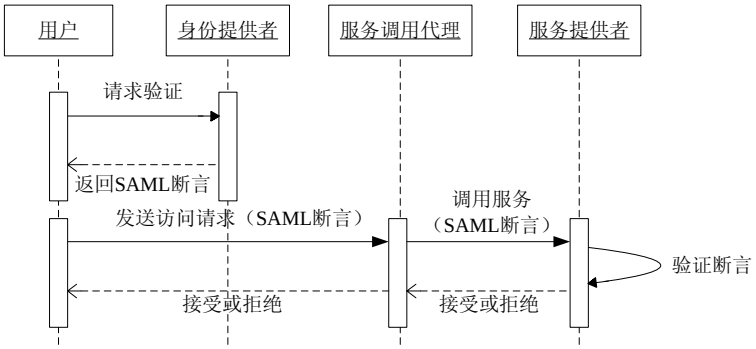

图 3-3 单点登录 Push 模式

3.3.2 Pull 模式

Pull 模式是指服务提供者从身份提供者那把断言拉过来。在 Pull 方式中，SAML 声明保存在身份提供者方，当服务提供者需要时，才从身份提供者方拉回此 SAML 声明。所以在用户向服务提供者方提出的请求中，只包含了一个对 SAML 的引用，这种引用用助诊文件（Artifact）来表示。Artifact 是 Base64 编码的字串,总长 42 字节,其中前 2 字节为类型码（Type Code）,后 40 字节中前 20 字节为源站点 ID,后 20 字节为声明的头信息，用来标识 SAML 声明。所以 Pull 方

式的认证过程相对复杂一些，具体的执行步骤如图 3-4 所示：

1) 用户先请求身份提供者，输入自己的身份信息。

2) 身份提供者给用户返回一个 SAML 助诊文件。

3) 用户把服务访问请求发给服务调用代理，请求中含助诊文件。

4) 服务调用代理调用具体的服务。

5) 服务提供者向身份提供者请求断言（请求消息中含助诊文件）。

6) 如助诊是合法的，则返回 SAML 断言；否则拒绝请求。

7) 服务提供者根据 SAML 断言来决定是否接受用户的请求，返回给服务调用代理。

8) 服务调用结果返回给用户。

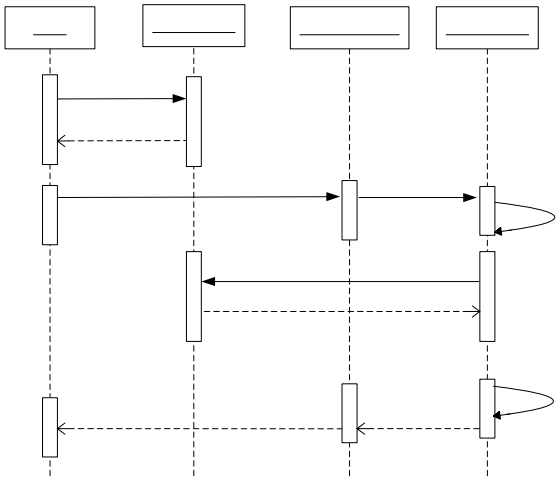

图 3-4 单点登录 Pull 模式

3.3.3 Pull 和 Push 的比较

Push 模式相对 Pull 模式较为简单，但断言部分是从客户端传向服务提供者的，所以安全性有一定的限制。而 Pull 模式中客户端只有助诊文件，助诊文件只是对验证断言的一个引用，真正验证是发生在身份提供者和服务提供者之间。而且身份提供者在向服务提供者提供断言的同时，会把助诊文件与断言的映射关系删除，也就是说助诊文件只能使用一次，防止了重放攻击。所以说 Pull 模式安全性更好一些，具体使用哪一种模式，要根据具体的环境来确定。

3.3.4 基于 Push 模式实现组合服务的单点登录

服务提供者提供的服务既可以是单个服务，也可以是组合服务。当为组合服务时，其中各成员服务可能也需要对用户进行验证。此处不考虑成员服务已把自己的验证与授权控制公开给组合服务提供者，由组合服务提供者来对用户进行验证，用户通过组合服务的验证，就可以访问所有成员服务的资源。因为成员服务可能考虑到安全问题，并不会把验证和授权委托给组合服务提供者，而是由自己去验证用户。对于成员服务把验证和授权委托给组合服务提供者去处理的场景，可以把组合服务当做单个服务来处理。以下讨论的是各成员服务分别对用户进行身份认证的场景。

当是单个服务时，使用 Push 和 Pull 模式都可以实现单点登录。当为上述场景的组合服务时，单点登录模型需要用 Push 模式来实现。因为在 Pull 模式中，助诊文件只能使用一次，但组合服务中每个服务都需要对用户身份进行验证。当采用 Pull 模式时，如果组合服务中有 n 个服务，则需要在身份提供者模块中相应的验证 n 次。所以，使用 Push 模式来实现组合服务的单点登录更恰当一些。当然，采用 Push 和 Pull 的组合模式也可以实现组合服务的单点登录，但比较复杂，本文对此种情形不做研究。

组合服务中的单点登录过程如图 3-5：用户从服务提供者获得断言后，传给服务调用代理，服务调用代理调用具体的组合服务，再由组合服务执行引擎按照流程依次调用各成员服务。

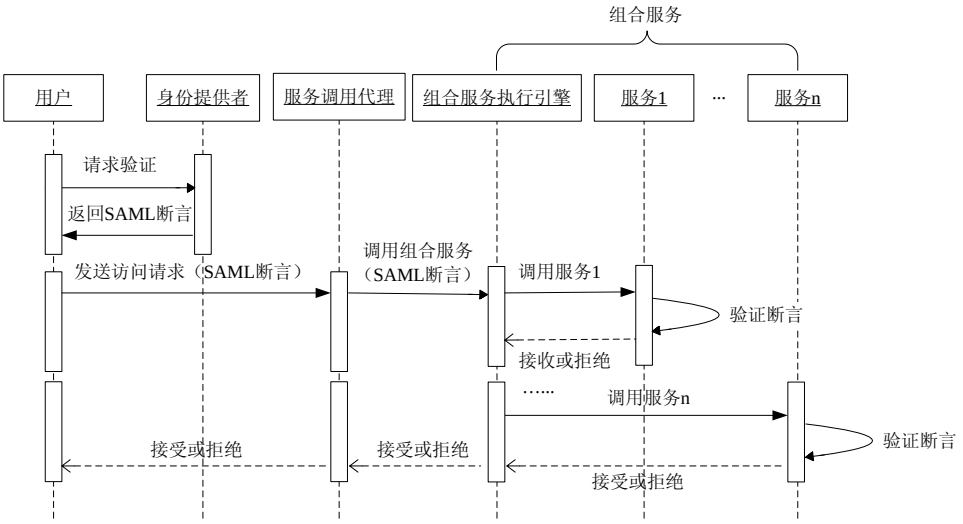

图 3-5 组合服务中的单点登录

3.4 示例

A 医院的一医生张某想使用一个遗传病决策支持服务，这个服务可以查亲属关系,病人及亲属的病史，最后进行决策。因为现在没有一个单服务可以提供这样的功能，但可以将多个服务组合起来完成该业务功能。A 医院提供一个组合服务，分别组合了亲属关系查询服务、病史查询服务、疾病决策支持服务，这些服务既可以是具体的一些服务，也可以是抽象的服务，即在服务调用过程中动态的选取具体的服务。现假设 A 医院的遗传病决策支持组合服务动态组合了区政府提供的亲属关系查询服务（Genetic Relation Query Service,简写为 GRQS），社区医疗机构提供的病史查询服务（Disease History Query Service,简写为 DHQS），B 医院提供的疾病决策支持服务（Disease Decision Support Service,简写为 DDSS）。组合服务的执行流程如图 3-6 所示：组合服务执行引擎先调用 GRQS,获得病人家属信息；然后调用 DHQS，获得病人家族的病史；最后调用 DDSS，进行决策，并把决策结果返回给服务调用者。

图 3-6 组合服务执行流程

张医生首先须在其自己单位提供的身份认证系统中登录，并获得身份认证系统的断言。断言格式如图 3-7 所示,主要由验证断言陈述 AuthenticationStatement 和属性断言陈述 AttributeStatement 组成。

张医生在访问服务时，将使用获得的断言来供服务提供者验证和授权。访问过程如图 3-8 所示，张医生在其自己单位类别的身份管理系统中获取断言后，发送访问请求给服务调用代理，服务调用代理调用组合服务，再由组合服务执行引擎依次调用各服务。

3.5 模型安全性分析

在 Web 服务单点登录过程中需要考虑传输的安全性，包括消息的保密性、完整性、不可抵赖性，重放攻击、中间人攻击以及断言的安全等。文献[56][57]对基于 SAML 的 Web 应用的单点登录进行了安全分析，证明了通信过程中一直使用 SSL/TLS 的安全通道就可以解决大部分安全问题。但 Web 服务环境有其自己的特点，SOAP 消息在传输过程中可能存在中间结点，SSL/TLS 虽然能保证点到点的安全问题，但并不能保证端到端的安全[58]。所以必须还要结合 XML Signature、XML Encryption、WS-Security 这三个规范来保证端到端的安全。

1. 保密性

保密性意味着只有期望的接收方才能读取消息的内容,而在消息发送途中截取消息的其他任何人都不能读取它的内容。在上述模型当中，一方面考虑传输过程的机密性,传输过程中都使用 SSL 3.0/TLS 1.0。另一方面由于消息可以被中间结点获得，所以需要对消息本身进行加密,可以使用 XML 加密技术选择性地对这些部分进行加密。

```
<saml:Assertion AssertionID="cT_S_T-vKMwidT8_Pzkke8UkC68."
IssueInstant="2008-02-25T16:31:03Z" Issuer="http://aaremote.entegrity.com">
<saml:Conditions NotBefore="2008-02-25T16:26:03Z"
NotOnOrAfter="2008-02-25T16:36:03Z"/>
<saml:AuthenticationStatement AuthenticationInstant="2008-02-25T16:30:58Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:am:password">
<saml:Subject><saml:NameIdentifier>uid=doctor zhang </saml:NameIdentifier>
<saml:SubjectConfirmation>
<saml:ConfirmationMethod>
urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
</saml:ConfirmationMethod>
<ds:KeyInfo>
<ds:KeyValue> doctor zhang ‘s public key</ds:KeyValue>
</ds:KeyInfo>
</saml:SubjectConfirmation>
</saml:Subject>
</saml:AuthenticationStatement>
<saml:AttributeStatement>
<saml:Subject>……</saml:Subject>
<saml:Attribute AttributeName="departmentclassclass">
<saml:AttributeValue>hospital</saml:AttributeValue>
</saml:Attribute>
……
<saml:Attribute AttributeName="title" >
<saml:AttributeValue> low</saml:AttributeValue>
</saml:Attribute>
</saml:AttributeStatement>
</saml:Assertion>
```

图 3- 7 断言示例

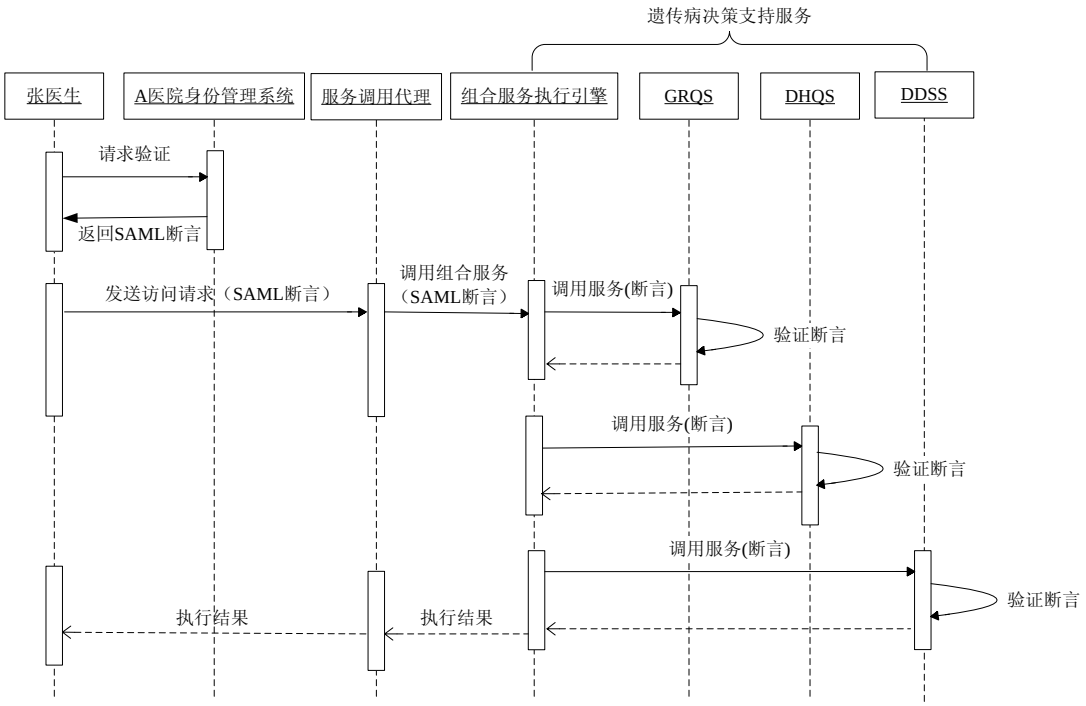

图 3-8 单点登录过程

2. 完整性和不可抵赖性

模型中对 SAML 的断言或助诊都使用了数字签名，可以防止消息被篡改以及抗抵赖。另外用户与身份提供者、服务提供者之间的通信信道采用了 SSL 3.0/TLS 1.0,身份提供者和服务提供者之间的通信采用了双向的安全通道。所以可以保证消息的完整性和不可抵赖性。

3. 重放攻击

由于 SOAP 消息及 SSL3.0/TLS1.0 中都加入了时间戳，接收方对消息头中的时间戳进行检测，从而避免重放攻击。

4. 中间人攻击

由于上述模型的所有通信过程都基于 SSL3.0/TLS1.0，所以中间人即使偷窃到了消息，但由于消息是经过加密和签名的，所以中间人也很难得到明文信息以及修改明文信息。

5. 拒绝服务攻击

上述模型中，身份提供者构造断言、服务提供者验证断言、服务提供者向身份提供者断言查询等过程都是十分耗时的，极有可能发生拒绝服务攻击。可以要求请求方对消息进行数字签名，比如服务提供者向身份提供者发送 SAML Request 时，必须对 SAML Request 进行数字签名。由于签名的耗时比确认签名

的耗时少，所以一定程度上缓解了拒绝服务攻击。

6. 断言的安全

断言在整个单点登录中处于核心地位，任何一个客户端拿到了断言，都可以进行访问目标服务。所以必须要对断言进行保护，以防被攻击者获取。Pull 模式是一种较好的方式，断言不经过客户端，对于单服务可以较好的解决断言安全问题。但在组合服务中，由于助诊只能使用一次，所以并不能适用，反而 Push 模式更适合组合服务。模型在所有通信过程中都使用了安全通道，断言相对是安全的，所以只要保证断言在客户端本地是安全的，则整个单点登录过程就是相对安全的。另外身份提供者在生成断言或助诊文件时应加上尽可能短的时间条件限制，来确保这两种文件的有效期。

如果客户端用户拥有一对公私钥，上述断言的安全问题则可以得到解决。断言中采用 holder-of-key 方法来验证主体的身份，即断言中包含了主体的公钥，当客户端发送访问请求时，用自己的私钥签名一段信息，服务提供者在验证断言时即可判断用户的身份。即使断言被攻击者获取，攻击者也无法使用断言来访问目标服务。

综上所述：该模型的安全性较好，可以适用于跨域的 Web 服务单点登录场景。

3.6 本章小结

基于 SAML 标准，本文设计了 Web 服务单点登录模型，给出了单点登录的 Push 和 Pull 模式的实现方法，对这两种模式进行了比较，并基于 Push 模式实现组合服务的单点登录，并以一个示例来说明单点登录的过程。最后对模型的安全性做了分析。模型的安全性较好，可以适用于 Web 服务环境下的单点登录。

第四章 一种具有协商机制的 Web 服务访问控制模型

4.1 引言

Web 服务是一种完全基于 XML 的软件技术，具有松散耦合、语言中立、平台无关等特点。企业可以把软件封装成 Web 服务，以提供给合作伙伴使用。Web 服务的实现遵循了一系列开放标准协议的，比如 WSDL、UDDI、SOAP、HTTP、SMTP 等。这些开放的标准协议使得 Web 服务很容易受到安全性的威胁，同时由于它的松散耦合性使得传统的安全机制也不能保证 Web 服务的安全性。访问控制是信息安全领域中的一个基础性的核心问题，因此，研究 Web 服务环境下的访问控制具有十分重要的意义。

本章提出了一种基于属性的具有协商机制的 Web 服务访问控制模型（Nego-ABAC），此模型继承了基于角色的访问控制模型（RBAC）的所有特点，如最小特权、责任分离和数据抽象等。同时，Nego-ABAC 引入了用户属性概念，避免了角色扩散；引入了上下文参数（资源属性和环境属性），使得模型是上下文可知的。当访问请求与策略不匹配时，不是直接的拒绝请求，而是在下文提到的部分接受情况下，通过与用户协商，让其在请求中修改一些参数和属性，使得服务的访问控制机制更加灵活。本章给出了模型的形式化描述、访问控制规则、完整的协商机制（包括协商准则以及协商消息格式）、整个访问控制的算法等。并给出了基于 XACML 的 Nego-ABAC 实现结构及 Nego-ABAC 中的策略与 XACML 策略的映射关系。

文献[28]提出了一种带有协商机制的 ABAC 模型，可以协商服务的参数，本文在其基础上进一步扩展，加入了对用户属性进行协商，即服务提供者通过协商来获得服务请求者未提供的一些属性。文献[30][31]提出的 WS-ABAC 模型，结合了自动信任协商技术，主要侧重点在于建立信任，保护敏感属性。而本文提出的协商，在于提高服务请求授权的自治性，用户通过动态调整参数和属性来适应服务提供者的访问策略。文献[29][32-36]提出的基于属性的 Web 服务访问控制都没有提供协商机制，本文提出的 Nego-ABAC 模型的自治性优于它们。

4.2 Nego-ABAC 模型

4.2.1 访问请求和访问控制策略的形式化描述

为了描述 Nego-ABAC 模型，本文在文献[28]的基础上做了一些扩展，给出了模型中的各部分形式化定义如下（服务提供者与身份提供者第三章已有定义，此处不再定义）：

定义 4- 1 用户属性：一个用户属性 Attr 是一个（SA,SA-domain）的二元组，其中 SA 代表该属性的名称，SA-domain 是该属性 SA 的值域。该值域可以用两种形式表示，一种是离散的，如$\{value_1, value_{2,...,} value_n\}$,另一种是连续的$[value_{begin},value_{end}]$。

定义 4- 2 资源属性：一个资源属性 Attr 是一个（RA,RA-domain）的二元组，其中 RA 代表该属性的名称，RA-domain 是该属性 RA 的值域。该值域可以用两种形式表示，一种是$\{value_1, value_{2,...,} value_n\}$,另一种是 $[value_{begin},value_{end}]$。

定义 4- 3 上下文环境属性：一个用户属性 Attr 是一个（EA,EA-domain）的二元组，其中 EA 代表该属性的名称，EA-domain 是该属性 EA 的值域。该值域可以用两种形式表示，一种$\{value_1, value_{2,...,} value_n\}$,另一种是 $[value_{begin},value_{end}]$。

定义 4- 4Web 服务参数：一个 Web 服务参数 Par 是一个（P,P-domain）的二元组，其中 P 是参数名称，P-domain 是该参数 P 的值域。该值域也可以有两种形式表示，一种是$\{value_1, value_{2,...,} value_n\}$，另一种是$[value_{begin},value_{end}]$。

定义 4- 5 访问请求：一个访问请求 Acc 是一个三元组$<\overline{sa},S,\overline{p}>$,其中：

1. $\overline{sa}=[SA_1:sa_{1,...,} SA_n:sa_n]$, SA_i 是用户属性名，sa_i 是属性 SA_i 相应的值；
2. S 是 Web 服务标识，唯一标识网络上的一个服务；
3. $\overline{p}=[P_1:p_{1,...,}P_k:p_k]$，$P_i$ 是服务参数名，p_i 是参数对应的值，该值须在 Web 服务参数对应的值域中。

定义 4- 6 属性条件集：$CASet=\{ca_1, ca_{2,...,} ca_n\}$，$ca_i=[AT,C]$,其中：

1. AT（Attribute Type）是属性类型，包括用户属性（SA），资源属性（RA），环境属性（EA）；
2. C（Condition）是属性条件，形如 A op k, C 的值为 true 或者 false。其中，A 是属性名，op 是一个比较操作符,例如≠、＜、＞、=、≤、≥，k 是一常量，且 k∈A-domain 中。

定义 4-7 访问控制策略：一个访问控制策略是一个三元组，形式为 pol=<S,Subset-par; CASet >,其中：

1. S 是 Web 服务标识，唯一标识网络上的一个服务；
2. Subset-par=$[P_1:subsetP_1,\ldots,P_k:subsetP_k]$,其中 P_i（i=1…k）是服务 S 的参数

名，subsetP_i 是 P_i-domain 的一个子集；

3. CASet 表示该策略需要的属性条件集。

4.2.2 访问控制规则

基于上述形式化定义，给出了 Nego-ABAC 模型中访问控制的一些规则。主要是策略吻合、完全接受及部分接受，首先访问请求与策略必须是吻合的，然后判断是完全接受还是部分接受。如果是完全接受，则允许访问；如果是部分接受，则需要协商；如果两者都不是，就拒绝请求。

规则 4-1 策略吻合：设 acc=<$\overline{sa}$,S,$\bar{p}$>是一个访问请求，pol=<S,Subset-par,CASet>是一个访问控制策略，如果满足以下几个条件，则称该请求与该策略是吻合的，即策略吻合。

条件1. acc.S=pol.S,即访问请求中的服务与策略中对应的服务一致；

条件2. $\forall$CASet. ca_i, i=1,…,n（CASet. ca_i. AT= RA or EA $\rightarrow$CASet. ca_i.C）。即条件集中的资源属性条件限制和环境属性条件限制必须要满足。

规则 4-2 完全接受：设 acc=<$\overline{sa}$,S,$\bar{p}$>是一个访问请求，pol=<S,Subset-par; CASet>是一个访问控制策略，如果满足以下几个条件，则称该访问请求被该策略完全接受。

条件1. acc 与 pol 必须是策略吻合的；

条件2. $\forall$ pol.Subset-Par.P_i, i=1,…,n,$\exists$j, j=1,…k,（pol.Subset-Par.P_i=acc.$\bar{p}$.P_j Λ acc.$\bar{p}$.$p_j\in$ pol.Subset-Par.subsetP_i），即访问请求的参数所取值在策略定义的参数范围内；

条件3. $\forall$ CASet. ca_i， i=1,…,n （CASet. ca_i.. AT=SA $\rightarrow$ $\exists$ j， j=1,…k（acc.$\overline{sa}$.SA_j= CASet.Ca_i..C.A Λ CASet.Ca_i..C（把 acc.$\overline{sa}$.sa_j 赋给 A））），即策略中的每个用户属性限制，访问请求中提供的属性值都满足。

规则 4-3 部分接受：设 acc=<$\overline{sa}$,S,$\bar{p}$>是一个访问请求，pol=< S,Subset-par; CASet>是一个访问控制策略，如果 acc 与 pol 是策略吻合，但非完全接受,且不符合下面这个条件，则称访问请求被该策略部分接受。

条件1. $\exists$CASet. ca_i, i=1,…,n（CASet. ca_i..AT = SA Λ $\exists$j, j=1,…k（acc.$\overline{sa}$.SA_j= CASet.Ca_i..C.A Λ $\neg$CASet.Ca_i..C））。即策略中规定的某些属性，用户请求中提供了，但不符合策略中属性条件，访问请求需要拒绝。

规则 4-4 访问控制：设 acc=<$\overline{sa}$,S,$\bar{p}$>是一个访问请求，pol=< S,Subset-par; CASet>是一个访问控制策略。如果 acc 被 pol 完全接受，则允许访问服务；如果部分接受，则可以使用协商机制来调整请求，直到完全接受时，才可以访问服务；否则，则拒绝请求。如果一个访问请求，对应多个策略时，可以有多种处理算法，

本文采取的是只要有一个策略满足，则接受该请求。

4.2.3 协商机制

当访问参数不符合访问策略中的定义，及用户没有提供足够的属性（排除提供了属性，但与策略不一致的情况）时，请求是部分接受的。为了提高访问控制的自适应性，给出协商建议，使用户可以调整访问参数和属性，来适应访问控制策略。协商建议的生成规则如下：

规则 4-5 生成协商建议：设 acc=<$\overline{sa}$,S,$\overline{p}$>是一个访问请求，pol=<S, Subset-par,CASet>是一个访问控制策略。如果 acc 被 pol 部分接受，则给出协商建议 Neg-Sug=<NegId,S, NegResultSet>,其中：

1. NegId 是在服务提供者方生成的唯一的一个标志符，用来关联客户端的第一次访问请求，客户端再次访问时只需提供协商所需的属性和参数及 NegId 即可；
2. S 是 Web 服务标识符；
3. NegResultSet 中包括一系列的协商内容，可表示为{$NegResult_1$,…, $NegResult_n$}, NegResult=<NegType, NegValue>,其中：

1) NegType 可取两种值，一种为 ParaDisMatch，服务参数不匹配，另一种为 SubAttrNotEnough，用户属性不充足；

2) 当 NegType=ParaDisMatch 时，NegValue=pol.Subset-par_i，即返回策略中所需的服务参数，包括参数名称和值域；当 NegType=SubAttrNotEnough 时，NegValue=CASet.Ca_i.C.A，即返回需要的属性名。

为了使协商过程正确有效的执行，给出如下协商规则：

规则 4-6 协商准则：因为每个部分接受的策略都可能会返回一个 Neg-Sug，所以对于多策略情况返回给用户的可能是一个协商建议集合。用户可以在协商建议集中选择一个 Neg-Sug 进行协商，构造协商响应 NegResponse（消息格式如规则 4-7）。服务提供者方在评估前首先进行如下两点的判断：

1. 协商响应中的服务参数若还不满足策略，则结束协商，拒绝请求；
2. 协商建议中要求提供的属性用户仍没有提供，则结束协商，拒绝请求。

若上述两点都不成立时，则开始评估，若是完全接受，则允许访问服务；若是部分接受，则继续协商，生成协商建议等，直到接受或拒绝请求；否则拒绝访问服务。

协商的过程如图 4- 1 所示：

1) 对于没有提供的属性，通过属性查询自动来实现，身份提供者提供了服务（属性权威）来供服务提供者查询用户属性。

2) 身份提供者进行敏感属性评估，如果是非敏感的属性，则提供给服务提供者，否则不提供属性给服务提供者，具体处理将在下章详细讨论。

3) 服务提供者按照规则 4-5 和 4.2.4 节中的算法 2 EvaluateNego()来生成协商建议，把不匹配的参数及没有查询到的属性都返回给用户。

4) 若用户接受协商，则调整服务参数，并修改敏感属性保护策略（见下章），构造协商响应，传递给服务提供者。

5) 服务提供者处理协商响应（包括对参数的处理以及属性的查询），并再次评估以确定是接受请求，还是拒绝，及继续协商。如果是继续协商，则循环到第三步重新处理。

6) 做出访问控制决定，把结果返回给用户。

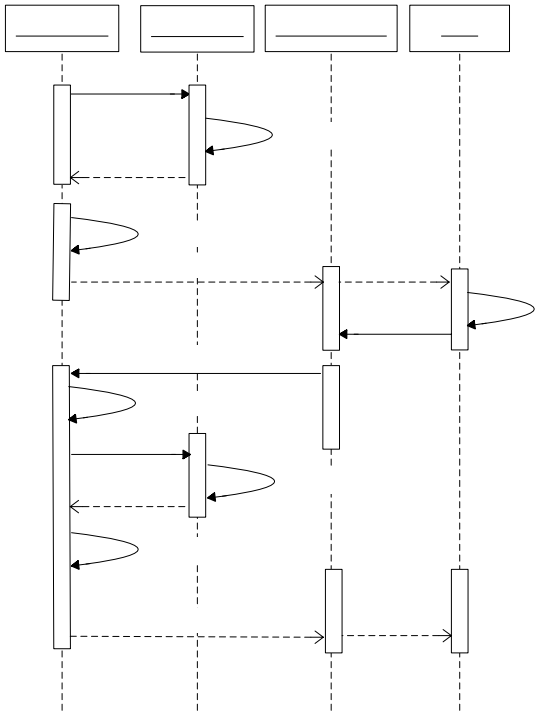

图 4- 1 协商过程

服务提供者 身份提供者 服务

再次访问时请求的格式如规则 4-7 所示。

请求属性

规则 4-7 再次访问请求消息格式：RequestAgain=< NegId,S,NegResponseSet>, RequestAgain 与 Neg-Sug 中的 NegId、S 相同，NegResponseSet={$NegResponse_1$,…, $NegResponse_n$}，NegResponse=<NegType,NegValue>,NegType=ParaDisMatch 或者是 SubAttrNotEnough，NegValue=<Name,Value>,当 NegType=ParaDisMatch 时，NegValue 分别为参数名和值；当 NegType=SubAttrNotEnough, NegValue 中的 Name 为属性名，Value 为 true 或 false，表示已提供或未提供。

敏感

生成协商建议

协商建议

再次请求

4.2.4 访问控制算法

基于上述形式化定义、访问控制规则及协商机制，给出了以下几个访问控制算法。算法 1 访问控制是总控程序，用来接受请求并进行访问控制。它调用算法 2 来评估属性，当部分接受时生成协商建议；调用算法 3 是对用户的协商响应进行处理。其中算法 3 是一个递规程序，根据规则 4-6 协商准则，当协商响应不符合一定的条件，就会拒绝协商。所以说该算法不会出现死循环，递规的深度是有限的。

算法1. 访问控制 AccessControl()

INPUT:

acc=<$\overline{sa}$,S,$\bar{p}$>//对于服务 S 的一个访问请求

POL={pol_1, pol_2,..., pol_n}//策略集合

OUTPUT:

Accept: 请求接受

Reject：请求拒绝

Begin function

POL'=Φ

//查找策略

for each pol_i∈ POL do

if pol_i.S=acc.S then

POL'= POL'∪{ pol_i}

end if

end for

if POL'=Φ then

return reject //没有相关的策略集

else

NegotiatedSugSet =Φ //协商建议集合

// NegotiatedSugSet ={neg-sug_1,..., neg- sug_k}，neg-sug_i(i=1,…k)是其中某个策略对该请求的协商建议

for each pol_i∈ POL' do

eResult= EvaluateNego(acc, pol_i)//调用算法 2

if eResult.AcceptLevel==fully then //完全接受

return Accept //如果有一个策略满足，则接受该请求

else

if eResult. AcceptLevel==partial then //部分接受
neg-sug = eResult.Neg-Sug //协商建议
NegotiatedSugSet = NegotiatedSugSet∪{neg-sug}
end if
end if
end for
end if
if NegotiatedSugSet ==Φ then
return Reject//协商集合为空，说明所有策略都拒绝了该请求
else//开始协商
SendNegoSugToUser(NegotiatedSugSet)//发送协商建议给用户
RequestAgain =GetNegoResp()//获得协商响应
return Negotiation(RequestAgain)//协商处理，调用算法 3
end if
End function

算法2. 评估请求及协商建议生成：EvaluateNego()

INPUT:
acc=<$\overline{sa}$,S,$\bar{p}$>//对于服务 S 的一个访问请求
pol=<S,Subset-par,CASet>//对应的一个访问控制策略
OUTPUT:
Result //评估结果，Result=<AcceptLevel, Neg-Sug>，AcceptLevel 的取值范围是{fully,partial,reject}；Neg-Sug 生成的格式如规则 4-5, Neg-Sug=<NegId,S,NegResultSet>,NegResultSet={$NegResult_1$,…, $NegResult_n$}，NegResult={NegType, NegValue},NegType=[ParaDisMatch, SubAttrNotEnough]
Begin function
Result.AcceptLevel=fully//默认是完全满足
Result.Neg-Sug=Null
for each pol.CASet.ca_i do//处理策略中的属性条件
if pol.CASet.ca_i..AT==RA or EA&&!pol CASet.ca_i..C then
Result.AcceptLevel=reject//资源和环境属性不满足，拒绝请求
return Result
end if
if pol.CASet.ca_i..AT ==SA then

if IsExistInAR(pol.CASet.ca_i.C.A) then//策略中的该属性名在访问请求中存在

if !pol.CASet.$ca_{i.}$.C then

Result.AcceptLevel=reject//用户属性不满足直接拒绝

return Result

end if

else //请求中没有该属性

if GetFromIdP(pol.CASet.ca_i.C.A) !=Null then

//从身份提供者处取属性

ReEvaluate();//取到属性后重新评估

else//取不到属性

Result.AcceptLevel=partial

NegResult=<SubAttrNotEnough, pol.CASet.ca_i.C.A>

NegResultSet= NegResultSet$\cup${NegResult}

end if

end if

end if

end for

for each pol.Subset-Par.P_i do//处理策略中的参数条件

if !IsExistInAQ(pol.Subset-Par.P_i) or acc.$\bar{p}$.$p_j \notin$ pol.Subset-Par.subsetP_i then//请求中不存在该参数或参数范围不正确

Result.AcceptLevel=partial

NegResult=<ParaDisMatch, pol.Subset-par$_i$>

NegResultSet=NegResultSet$\cup${NegResult}

end if

end for

if Result.AcceptLevel==partial then

Result. Neg-Sug. NegResultSet= NegResultSet

end if

return Result

End function

算法3. 协商处理：Negotiation()

INPUT:

RequestAgain =< NegId,S,NegResponseSet>, NegResponseSet 的消息格式

如规则 4-7，NegResponseSet={NegResponse$_1$,…,NegResponse$_n$}，NegResponse={NegType,NegValue}，NegType 取值{ParaDisMatch, SubAttrNotEnough}，NegValue=<Name,Value>

```
OUTPUT:
Accept:请求接受
Reject:请求拒绝
Begin function
if NegIdMatch(NegId) then//协商 Id 的匹配
    if !ParameterMatch() then//参数仍然不匹配，规则 4-6 协商准则
        return Reject
    end if
    if GetAttribute()==Null then//获取不到属性，规则 4-6 协商准则
        return Reject
    end if
    if ReEvaluate(RequestAgain)==fully then//重新评估,与算法 2 类似
        return Accept
    else
        if ReEvaluate(RequestAgain)==reject then
            return Reject
        end if
    else
        if ReEvaluate(RequestAgain)== partial then
            SendNegoSugToUser(evaluationResult)//评估结果返回给用户
            RequestAgain =GetNegoResp()//获得协商响应
            return Negotiation (RequestAgain)//递规
        end if
    end if
else// NegId 不匹配
    return Reject
end if
End function
```

4.3 基于 XACML 的 Nego-ABAC 模型的实现结构

由于 XACML 规范充分考虑了主体、资源和环境的属性,而且还定义了一种

机制，用来创建进行授权决策所需的规则、策略和策略集的完整基础设施。所以本章基于它来实现 Nego-ABAC 模型。本节首先介绍模型的总体结构，然后介绍模型的执行机制，最后给出基于 XACML 的模型访问控制策略的表示。

4.3.1 总体结构

Nego-ABAC 模型可以基于用户、环境及资源的属性来进行访问控制授权，实现结构如图 4-2 所示，其中圆角矩形的是 XACML 规范中定义的一些组件，矩形是本模型扩展的组建。具体如下：

1.服务请求者：服务请求的发起方，它首先会向身份提供者请求验证自己的身份，获得断言后传给服务调用代理。

2.服务调用代理：为了使得服务对于用户的透明，它做为中介者来和服务提供者的组件交互。

3.身份提供者：对用户的身份进行确认，并发放断言，同时提供了属性查询功能，供策略信息点进行属性查询。

4.SOAP 处理器：用来接收 SOAP 消息，并确认消息中的数字签名，如果确认是合法的，则把 SOAP 消息体中的内容转换成一个访问请求，传递给策略执行点。

5.策略执行点：它把用户访问请求传递给上下文处理器，并根据上下文处理器的处理结果来执行相应的操作，比如拒绝访问或合法访问服务，同时会把给用户的响应结果传递给 SOAP 处理器。

6.上下文管理器：主要作用有以下几个方面：

1) 把策略执行点传递过来的访问请求转换成策略决策点可以识别的访问请求；

2) 从策略信息点处获得资源和环境的属性，传给策略决策点，以满足评估属性的需要；

3) 获得决策结果，传递给策略执行点。

7.策略决策点：根据用户属性，环境属性，资源属性，访问请求，以及策略管理点中的策略来判断用户是否可以访问相应的服务，具体的算法，如 4.2.4 中的算法 1 和算法 2。

8.协商决策点：在策略决策点做出结果是部分接受时，使用协商决策点来生成协商建议，调整一些参数和属性的，具体算法如 4.2.4 中的算法 2 协商部分，以及算法 3 协商处理。

9.策略管理点：制定策略并对策略进行管理。

10. 策略信息点：获取环境及资源属性，供策略决策点使用，并且可以从属

性权威查询相关的一些属性。

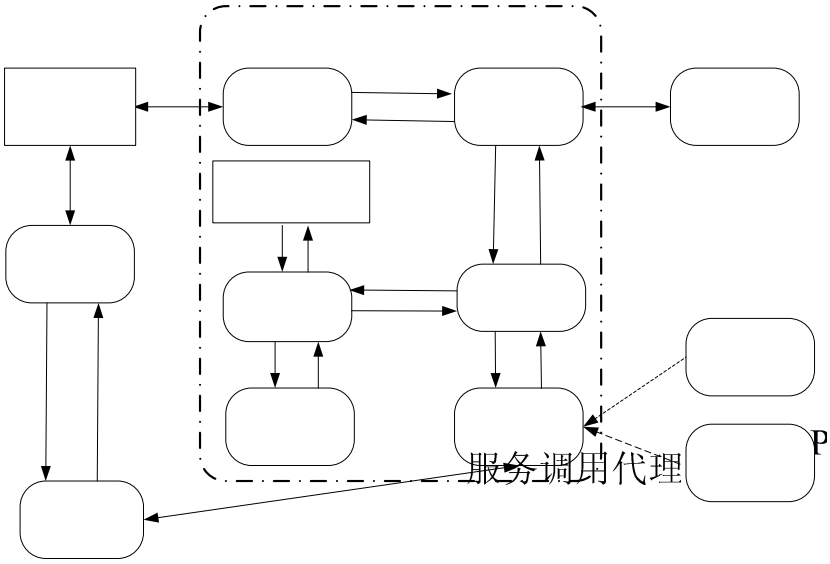

图 4-2 基于 XACML 的 Nego-ABAC 模型实现结构图

4.3.2 模型的执行机制

基于 XACML 的 Nego-ABAC 模型的执行机制如图 4-3 所示：

1. 访问请求者向身份请求者请求验证。
2. 身份提供者验证结果为合法用户，则发放断言，否则，退出。
3. 访问请求者向服务调用代理发送请求，由服务调用代理调用服务。
4. 策略执行点对请求消息进行处理，然后传给上下文处理器。
5. 上下文处理器把消息传递给策略决策点。
6. 策略决策点进行属性评估，并向上下文处理器进行属性查询。
7. 上下文处理器从策略信息点处获得资源、环境及主体属性。
8. 上下文处理器把获得的属性传递给策略决策点。
9. 策略决策点重新评估，如仍不符合，则向协商决策点发出协商请求。
10. 协商决策点处理协商请求，根据根据 4.2.4 中的算法 1 和 2 生成协商建议。
11. 协商建议返回到访问请求者。
12. 访问请求者重新调整访问参数和属性，再次构造请求。
13. 再次构造的请求传递到策略执行点时，策略执行点重新评估属性，根据 4.2.4 中的算法 3，决定是接受请求还是拒绝或继续协商。
14. 当允许访问服务时，策略执行点调用目标服务，并返回结果。

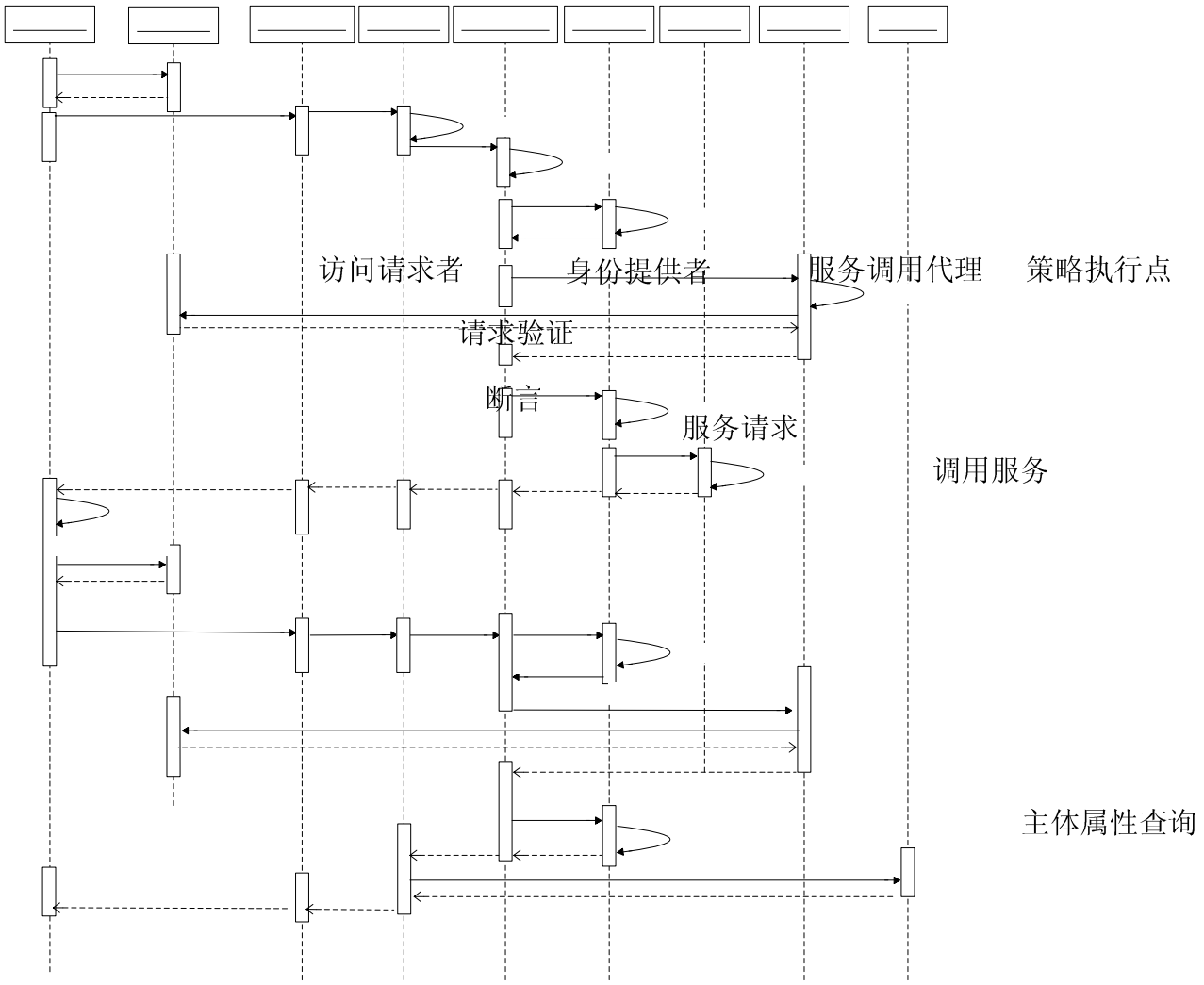

图 4-3 Nego-ABAC 模型执行机制

4.3.3 基于 XACML 的 Nego-ABAC 模型访问控制策略的表示

XACML 规范中定义了策略语言模型（policy language model）[55]，主要有规则（rule），策略（policy），策略集（policyset）。本章采用的 XACML 规范中定义的 only-one-applicable-policy 策略连接算法[55]，这个算法定义了策略集中只能有一个策略满足目标对象。在一个策略中，规则可以有多个，Nego-ABAC 模型中的策略 pol=<S,Subset-par; CASet >及策略集 polSet={pol_1,…, pol_n}与 XACML 中的规则和规则集对应。两者的对应关系如图 4-4 所示：

1. Nego-ABAC 模型策略 pol 中的 S 对应于 XACML Rule 中的 target 的 Resource 和 Action 两元素。

2. pol 中的 Subset-par 与 Rule 中的主体属性条件对应。

3. pol 中的 CASet 包括主体属性（SA）、环境属性（EA）及资源属性（RA）

条件,其中 SA 条件也与 Rule 中的主体属性条件对应，和 Subset-par 相同，两者之间用属性类型来区别。pol 中的 EA 条件与 Rule 中的环境属性条件对应。pol 中的 RA 条件与 Rule 中的资源属性条件对应。

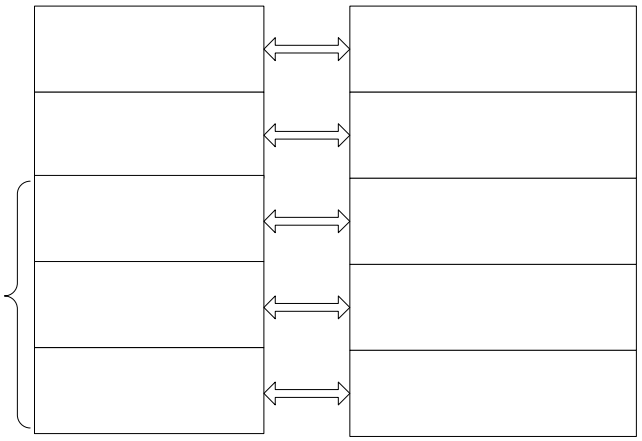

图 4-4 Nego-ABAC 中 pol 与 XACML 中 rule 对应关系

服务标识符S

Nego-ABAC 模型中的访问请求 Acc=<$\overline{sa}$,S,$\overline{p}$>与 XACML 中的访问请求对应关系如图 4-5 所示：

1. Acc 中的 S 与 Request 中的 Resource 和 Action 两元素对应。
2. Acc 中的$\overline{sa}$与 Request 中的主体属性对应。
3. Acc 中的$\overline{p}$和$\overline{sa}$一样，都用 Request 中的主体属性对应，两者用属性类型来区分。

参数集Subset-par

主体属性条件

Nego-ABAC 模型中的评估结果继承了 XACML 中的响应类型——允许、拒绝 [55]。同时添加了协商这种类型，使得访问控制系统具有更强的自适应性。

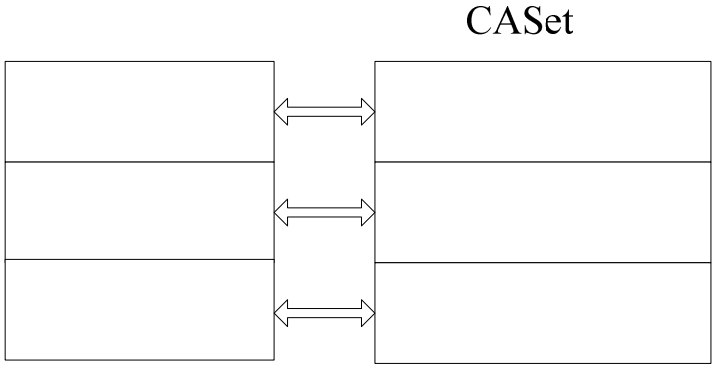

图 4-5 Nego-ABAC 中 Acc 与 XACML 中 Request 对应关系

4.4 模型安全性分析

本节接着第三章介绍的应用场景遗传病决策支持服务示例，介绍Nego-ABAC访问控制的过程，并对访问控制进行安全性分析。

病史查询服务（DHQS）为了保护服务被非法用户访问，制定了如下策略：

pol_1=< DHQS; Discount[0,50];{ departmentclass=hospital,title>=middle}>

pol_2=< DHQS; Discount[0,30];{ departmentclass=hospital, title<middle }>

策略中departmentclass、title为用户属性，分别表示用户的单位类别和职称。departmentclass的取值范围为{ hospital }，title的取值范围为{low,middle,high}。Discount为Web服务参数，表示查询费用的折扣，它的取值范围为[0,100]。pol_1的意思是当用户的单位类别是医院且职称是大于等于中级时，查询费用的折扣数在[0,50]。pol_2的意思是当用户的单位类别是医院且职称小于中级时，查询费用的折扣数在[0,30]。

如图4-6所示，张医生开始发送访问请求acc_1=<[departmentclass:hospital,title:low], DHQS, [Discount:50]>，他提供了单位类别和职称属性，并希望服务可以给予折扣50%。评估引擎对访问请求评估后，发现该请求的属性条件满足pol_2，但服务参数不符合。所以服务提供者返回协商建议neg-sug_1=<$negId_1$,DHQS,{< ParaDisMatch, Discount [0,30] >}>。张医生在收到协商建议后，如果接受协商，则返回协商响应neg-$response_1$=<negId,DHQS,{<ParaDisMatch ,<Discount, 30 >>}>。再次评估，接受用户请求。

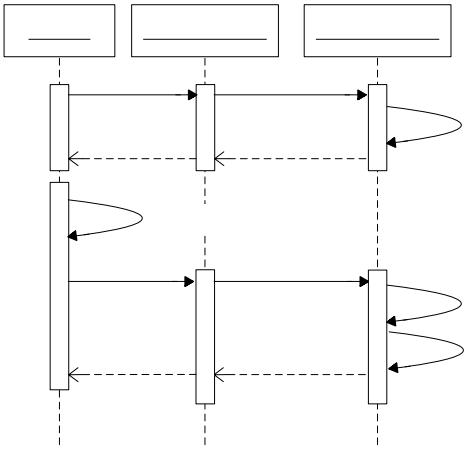

图4-6 示例一

如图4-7所示，如果张医生的访问请求是acc_2=<[departmentclass:hospital], DHQS, [Discount:50]>，他提供了单位类别属性，并希望服务可以给予折扣50%。评估引擎对访问请求评估后，返回协商建议{ neg-sug_2,neg-sug_3}，

neg-sug$_2$=<negId$_2$,DHQS,{<SubAttrNotEnough,title>}> 、 neg-sug$_3$=<negId$_3$, DHQS,{<SubAttrNotEnough, title>,<ParaDisMatch, Discount[0,30] >}>,张医生收到协商建议后，会选择其中一个协商建议进行协商。首先他发现 neg-sug$_2$ 满足他的参数需求，但需要职称属性。如果他接受协商，则需要修改参数并调整敏感属性保护策略，返回协商响应 neg-response$_2$=<negId,DHQS,{<SubAttrNotEnough,<title,true>>}>。DHQS 服务再次查询用户的属性，假设张医生的 title 为 low，重新评估后仍不被策略完全接受。但还有协商建议 neg-sug$_4$=<negId$_4$, DHQS,{ <ParaDisMatch, Discount[0,30] >}>存在，所以继续协商。张医生如果接受协商，则返回协商响应 neg-response$_3$=<negId,DHQS,{ <ParaDisMatch , <Discount, 30 >>}>。再次评估，接受用户请求。

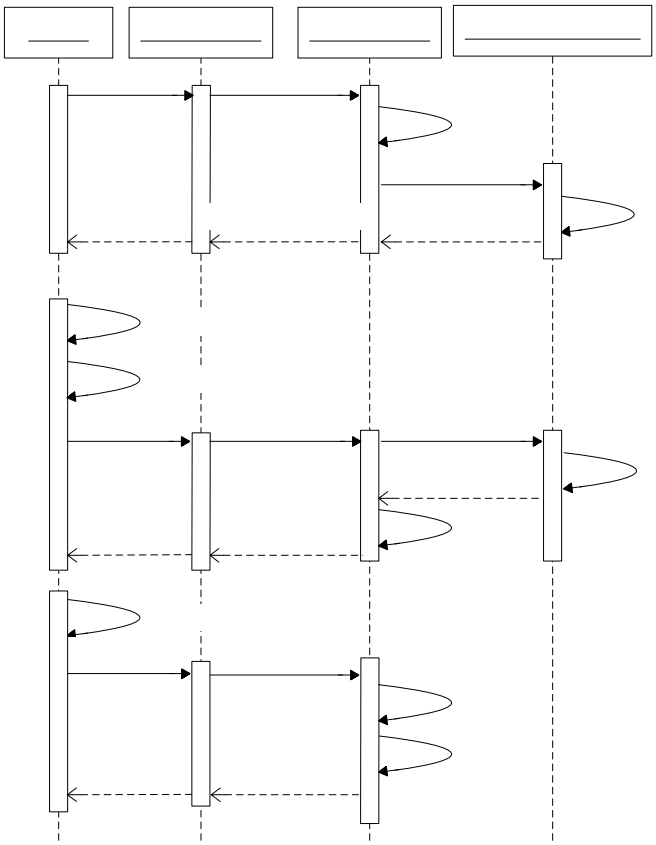

图 4-7 示例 2 张医生 服务调用代理 病史

基于属性的访问控制可以有效的对资源进行保护，只有满足属性条件，才可以访问相应的资源，实现了对资源的保护，安全性较好[28-31]。而 Nego-ABAC 在原有 ABAC 模型的基础上，加入了协商机制，并没有削弱访问控制，反而使得访问控制更加灵活，自治性更强。

根据 4.2.4 中的算法 1 计算访问，只有访问请求被策略完全接受时，才允许

acc2

neg-sug2和neg-su

访问资源，而部分接受或拒绝状态都不可以访问资源。同时在 4.2.4 中的算法 3 协商处理中，直到请求被完全接受时，才可以访问资源，否则继续协商或拒绝请求。所以 Nego-ABAC 保证了访问控制的安全性，使得不被控制策略完全接受的访问请求不能访问资源。

如示例 1 中，张医生的单位类别是 hospital、职称是 low，并希望得到 50%的折扣。但策略中定义单位类别为 hospital、职称是 low 的用户只能获得 30%以内的折扣。所以张医生若想获得服务，必须调整其参数，将折扣修改为 30%。同样如示例 2 中，张医生没有提供其职称属性，病史查询服务也无法从 A 医院身份管理系统中获得属性。根据 4.2.4 中的算法 1 和 2 服务提供者生成协商建议，返回给用户。用户修改敏感属性保护策略及调整服务参数，直到访问请求被访问控制策略完全接受，才可以访问服务资源。所以加入了协商，并没有降低访问控制的安全性，实现了增加访问控制自治性的目的。

4.5 本章小结

本章提出了一种基于用户、环境、资源属性的 Nego-ABAC 模型，给出了访问请求和访问控制策略的形式化定义、访问控制规则、协商机制、访问控制算法。给出了基于 XACML 的 Nego-ABAC 模型的实现结构、执行机制及 Nego-ABAC 模型与 XACML 两者的策略语言的对应关系。最后给出一个示例并分析了模型的安全性。Nego-ABAC 模型解决了传统的访问控制模型在 Web 服务环境下由于对用户身份不可知而导致的访问控制困难，以及角色扩散等问题，具有自治性和自适应性，适用于 Web 服务环境。

第五章 基于信任级别和协商机制的用户敏感属性保护研究

5.1 引言

基于属性的访问控制（Attribute-based access control, ABAC）机制比较适合于这种开放的 Web 服务环境[28-31]。在 ABAC 中，授权决策是基于请求者的属性（如年龄、资格等），而不仅仅是身份。这些属性通常用数字签名的凭证来建立，通过此凭证，凭证颁发者可断言某实体具有哪些属性。但这些属性有的是敏感的，需要加以保护，不是每个服务都可以访问用户的敏感属性。

本章提出了一种基于信任级别和协商机制的 Web 服务环境下的用户敏感属性保护模型。制定了一种 TLBAck 策略，策略中包含属性的信任级别，以及协商的条件。通过属性与服务的信任级别的比较，来决定是否披露敏感属性，简化了策略。加入协商条件，在特定条件下，可以通过协商来降低敏感属性的信任级别，从而使得敏感属性可被满足特定条件的服务访问，使得模型的自治性较强。信任级别和协商机制的结合使得对敏感属性保护的粒度更细，更加灵活，因为满足不同协商条件时敏感属性的信任级别是不同的，即使服务满足了这个协商条件，如果信任级别仍然低于敏感属性的信任级别，也是不可以访问敏感属性的。文中给出了模型的总体结构、形式化描述及算法，并对组合服务下的敏感属性保护进行了研究。

针对文献[37]提出的敏感属性保护方法的自治性差和只能应用于静态服务的不足，本文提出的用户敏感属性保护模型具有协商功能，且遵循了 SOC 思想，服务对于用户透明，服务的信任级别由第三方确定，而不是用户自己指定。文献[30]只给出了敏感属性保护的一个思路，而本文提出了较为完善的用户敏感属性保护模型，包括模型的总体结构、形式化描述、算法等，可以直接应用于各种场景。针对文献[47]中的 Ack 策略应用于 Web 服务环境时，策略非常复杂的不足，本文制定的 TLBAck 策略中引入了信任级别，可以较大程度的简化 Ack 策略，减少了用户的管理负担。

5.2 敏感属性保护模型

5.2.1 信息的敏感性

访问控制策略和属性凭证是属于 ATN 中的特殊资源。与一般的文件、服务

资源相类似，策略和属性凭证中也可能包含敏感信息。总的来说，目前所研究资源（主要是指访问控制策略和属性凭证）涉及的敏感信息不外乎两大类：

(1) 资源的内容敏感：访问控制策略（如实例 1），属性凭证中的某些属性值（如实例 2），属于显式敏感信息；

(2) 资源的拥有敏感：协商方的响应和信息流动会隐式的暴露其保密信息（如实例 3），属于隐式敏感信息。

实例1. A 公司制定了一项秘密联合计划，协定只有 B 公司的 CEO 和 C 公司的职员才能访问 A 公司的专用资源 R，访问控制策略为 $R \leftarrow (C_{B.}\text{title}=\text{"CEO"}) \vee C_{C,}$ C_B、C_C 为属性凭证。如果该策略公开，势必会暴露其合作的伙伴。

实例2. Alice 的驾照 C 属于交通部门颁发的属性凭证，但在某些情况下，Alice 并不希望将其中的年龄属性值 age 随意提供给陌生方，也就是说，C.age 属于用户的敏感属性。

实例3. Alice 属于某保密组织 S 的成员，拥有 S 为其颁发的属性凭证 C，如果单纯制定 C 的访问策略 P，Alice 在向 C 的请求方披露策略 P 或无响应时，都会隐式地暴露 Alice 是否拥有 C 的事实。

Seamons，Winslett 等人[59]从凭证角度划分出属性敏感凭证（attribute-sensitive credential）和拥有敏感凭证（possession-sensitive credential）两类，分别包含在上述的显式和隐式敏感信息分类范围内。Ack 策略既可以保护属性敏感凭证，又可以保护拥有敏感凭证。因为访问策略的泄漏问题比较容易解决，采取在任何情况下都不向外泄漏策略中的属性条件的值的方法，而只泄漏属性名称。所以在此不做讨论。

5.2.2 模型的总体结构

如图 5-1 所示，敏感属性保护模型是在上述单点登录和访问控制模型的基础上进行了一些扩展，并添加了一些模块。

身份提供者方主要扩展了发放断言和断言响应模块，并增加了敏感属性保护、协商策略及协商协议模块。

1) 敏感属性保护模块：基于下文中的定义 5-1TLBAck 策略来对用户的敏感属性进行保护。

2) 发放断言模块：身份提供者在发放断言时，会调用敏感属性保护模块，判断哪些属性是敏感属性。用户会对服务提供者的 TL 做一个限制，服务调用代理将不会去访问低于这个限制的服务。根据服务提供者的 TL 限制及用户自定义的属性信任级别，敏感属性保护模块计算出哪些属性披露给服务提供者。对于敏感的属性，将不会被加入到断言中，从而达到对敏感属性的保护。具体算法如

5.2.4 中的算法 1 敏感属性过滤。

3) 断言响应模块：对于属性查询的情况，也会调用敏感属性保护模块。对于敏感属性，不会披露属性，将采取协商策略，协商算法如 5.2.4 中的算法 2 和算法 3。

4) 协商策略模块：敏感属性在协商过程所采取的一些策略。

5) 协商协议模块：协商过程中信息交互的格式。

6) 属性凭证库：库中保存的是敏感属性的集合。

7) TLBAck 策略库：由主体制定的一些敏感属性保护策略集。

服务提供者方增加了属性协商、协商策略及协商协议模块。属性协商模块是用来接收身份提供者的协商请求，并给身份提供者返回属性凭证。协商策略及协商协议模块与身份提供者处类似，在此不再阐述。

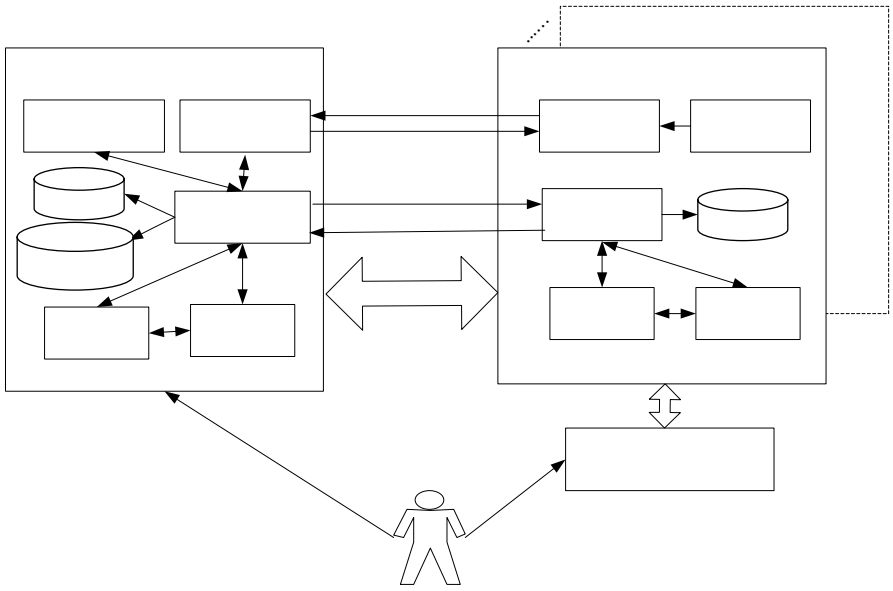

图 5-1 基于信任级别和协商机制的敏感属性保护模型图

如图 5-2 所示，模型的执行机制如下：

1) 用户制定 TLBAck 策略，并保存到身份提供者处。

2) 用户在向身份提供者请求断言时，依据 5.2.4 中的算法 1，身份提供者会根据敏感属性的信任级别及服务提供者的信任级别限制，来构造相应的属性断言 statement，敏感属性将不会放入断言中。

3) 用户向服务调用代理发送访问请求，由服务调用代理选取并调用具体的服务。

4) 服务提供者方的服务在访问控制评估中，如缺少属性，根据第四章的内容，它将会向身份提供者查询属性。

5) 身份提供者在调用敏感属性保护模块，发现此属性不能披露给服务提供

SAMLR

SAMLR

协商

敏感属性保护

属性

TLBAck策略

双向安全

者。所以身份提供者会将 NegCondition 披露给服务提供者。（按照 5.2.4 中的算法 2 和算法 3）

6) 服务提供者提供属性凭证给身份提供者，身份提供者确认并计算等级后，才决定是否把敏感属性发给服务提供者。如果属性凭证不满足 NegCondition，敏感属性协商失败。（按照 5.2.4 中的算法 2 和算法 3）

7) 访问控制模块重新评估，如果满足策略，则允许调用服务，并将服务调用结果返回。否则，会按照第四章中的协商机制来处理，返回协商建议。用户调整 TLBAck 策略，并返回协商响应，服务提供者再重新评估。

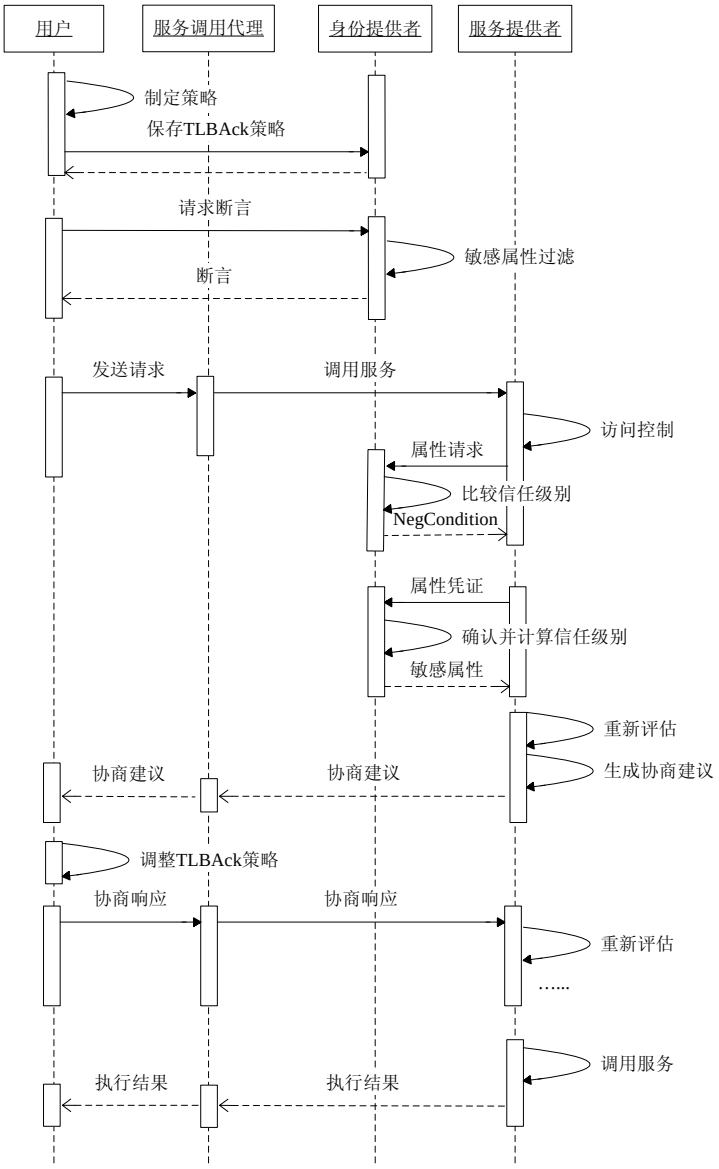

图 5-2 基于信任级别和协商机制的敏感属性保护模型执行图

5.2.3 形式化描述

定义 5-1 基于信任级别的响应策略（Trust Level Based Acknowledgement policies，简称 TLBAck 策略）：p=<u, ServiceTLConstraint,A>,其中：

1. u为用户标识；

2. ServiceTLConstraint标识服务的信任级别限制，服务调用代理只调用高于这个级别的服务；

3. A是用户u的一系列的属性集，$A=\{attr_1,attr_2,\ldots, attr_n\}$，其中，$attr_i$=<Name,TL,NegCondition>,Name为属性名称，TL为信任级别（假设已经有了一个信任机构对网络上所有的服务进行了信任级别的量化，所以用户可以给敏感属性加上阈值，只有服务的TL大于属性的TL,才可以访问该服务），NegCondition为协商条件。NegCondition={<$Condition_1$,TL_1>,<$Condition_2$,TL_2>,…,<$Condition_n$, TL_n>}，<$Condition_j$,TL_j>表示满足条件$Condition_j$时属性的信任级别。Condition是协商条件，形如A op k, C的值为true或者false。其中，A是属性名，op是一个比较操作符,例如≠、＜、＞、=、≤、≥，k是一常量。为了防止拥有敏感凭证泄漏，属性集中包含一些用户没有的属性。

定义 5-2 属性请求（Attribute Request，简写为 AR）：服务提供者向身份提供者发送的一个属性请求，AR=<u,S,A>,其中，u 为用户标识符号，S 为服务提供者，S=<SI,TL>，SI 为服务标识符，TL 为 S 的信任级别;A 为请求的属性集 $A=\{attr_1, attr_2,\ldots, attr_n\}$。

定义 5-3 用户属性的授权：WS 为一服务，$attr_i$ 为用户一属性，如果 $WS.TL \geq attr_i.TL$,则允许服务 WS 访问属性 $attr_i$，记为 WS_authorized_access_$attr_i$。

规则 5- 1 敏感属性过滤：在断言构造过程中，对敏感属性需进行过滤，属性断言陈述中的属性需满足如下条件：$\forall attr_i (attr_i.TL \leq \text{ServiceTLConstraint})$。

规则 5-2 属性披露协商过程：WS 为一服务，$attr_i$ 为用户一属性，如果 $WS.TL<attr_i.TL$，则可以通过协商来解决。用户向服务提供方依次披露 $p.A.attr_i$. NegCondition，服务提供方提供其属性凭证，再比较信任级别，判定是否可以访问该属性。

规则 5-3 属性披露协商策略：服务提供方可以拒绝提供属性凭证，协商失败并结束。如果服务提供方提供了属性凭证，但是仍不满足 TLBAck 策略，协商也为失败并结束。

规则 5-4 属性披露协商消息格式：身份提供者披露协商条件 NegCondDisclosure=<NegID, NegCondition >, NegID 是唯一生成的标识符，用来标识一次协商，NegCondition={<$Condition_1$,TL_1>,<$Condition_2$,TL_2>,…,<$Condition_n$,

TL_n>}。服务提供者披露凭证 AttrCredDisclosure=<NegID,NegValue>，NegValue=<AttributeName, AttributeValue>。

5.2.4 敏感属性保护算法

基于上述形式化描述，给出了如下三个算法：算法 1 是用户在身份提供方进行身份验证时，身份提供者对敏感属性进行了过滤，即生成的属性断言 statement 中不包含敏感属性；算法 2 是用户属性披露算法，决定用户请求中哪些属性可以披露，调用算法 3 进行协商，协商如果成功，则允许访问该属性；失败则拒绝访问该属性。

算法1. 敏感属性过滤 FilterSensitiveAttribute()

INPUT:PSet 用户策略集，u 用户标识

OUTPUT: AttributeAssertion Statement 属性断言陈述

Begin function

Assertion_Attr=Φ

Find $p_i \in P$ where u==p_i.u

if p_i==null then

return error//没有这个用户的策略

end if

for each $attr_k \in$ p.A do

if p.ServiceTLConstraint>= $attr_k$.TL then

Assertion_Attr= Assertion_Attr$\cup\{attr_k\}$

end if

end for

return MakeAttrAssertionStatement(Assertion_Attr)

End function

算法2. 用户敏感属性披露 SensitiveAttrDisclosure()

INPUT:AR(属性请求)，PSet 用户策略集

OUTPUT: S_Attr(服务可以访问的属性集) or error

Begin function

S_Attr=Φ

Find $p_i \in P$ where AR.u== p_i.u

if p_i==null then

return error//没有这个用户的策略

end if

for each $attr_k \in$ AR.A do
if WS_authorized_access_$attr_k$
S_Attr= S_Attr$\cup${$attr_k$}
else
if negotiation(p_i, $attr_k$)==success then
S_Attr= S_Attr$\cup${$attr_k$}
end if
end if
end for
End function

算法3. 协商 Negotiation()

INPUT: p(TLBAck 策略)，attr 属性
OUTPUT: success 协商成功 or fail 协商失败
Begin function
SendNegoCondition()//发送协商条件
GetSPAttribute()//获得服务提供者属性
if $Condition_i$ then//如果满足 TLBAck 策略中的条件
p.A.attr.TL= $Condition_i$.TL
end if
if WS_authorized_access_attr then// 重新判断信任级别是否满足
return success
else
return fail
end if
End function

5.3 组合服务情况下用户敏感属性保护

服务既可以是单个服务，又可以是组合服务。对于单个服务的情况，使用上述三个算法，即可解决敏感属性的保护问题。当是组合服务时，服务包含了一组子服务。组合服务中的子服务对于用户是透明的，是由服务执行引擎去依次执行这些子服务。处理的方法是：服务执行引擎在选取服务时，需要每个子服务的信任级别都满足用户定义的 ServiceTLConstraint，这样就不会发生敏感属性泄漏。

5.3.1 处理流程

对于组合服务的敏感属性保护处理流程如图 5-3 所示：

1. 用户向身份提供者请求断言,选取非敏感属性来构造断言。
2. 用户通过服务调用代理向组合服务执行引擎发送访问请求。
3. 组合服务执行引擎根据用户定义的 ServiceTLConstraint 来选取子服务。
4. 组合服务执行引擎依次调用各服务，当服务提供者还需要更多的属性时，则需要向身份提供者发送请求甚至需要协商等，处理如 5.2.4 中的三个算法以及第四章的访问控制算法。
5. 当调用第一个服务成功后，会按照流程的顺序调用其它服务，处理方式相同。
6. 当所有服务处理完毕后，才返回结果给用户。

5.3.2 应用示例

本节接着前面几章讨论的遗传病决策支持服务示例，来介绍敏感属性保护的过程。张医生现在不希望自己的一些属性被低 TL 的服务访问。所有他制定了 TLBAck 策略：p=<doctor zhang,3.0,{ <departmentclass,3.0,Φ>,<title,3.5>}>。因为用户定义 ServiceTLConstraint=3.0，所以身份提供者在构造属性断言时，包括了以下属性{ departmentclass}。

表 5-1 服务信任级别

	亲属关系查询服务（GRQS）	病史查询服务（DHQS）	疾病决策支持服务（DDSS）
TL	3.0	3.0	4.0

表 5-1 中是第三方提供的各服务的 TL 值，假定 TL 的值域是[0.0,10.0]。现假设服务执行引擎在调用 GRQS 时，该子服务需要{ departmentclass , title}这两个属性，首先 GRQS 会向 A 医院发送属性请求 AR=<doctor zhang, < GRQS,3.0>,{title}}>时，经过信任级别的比较，发现不能披露该属性给 GRQS。

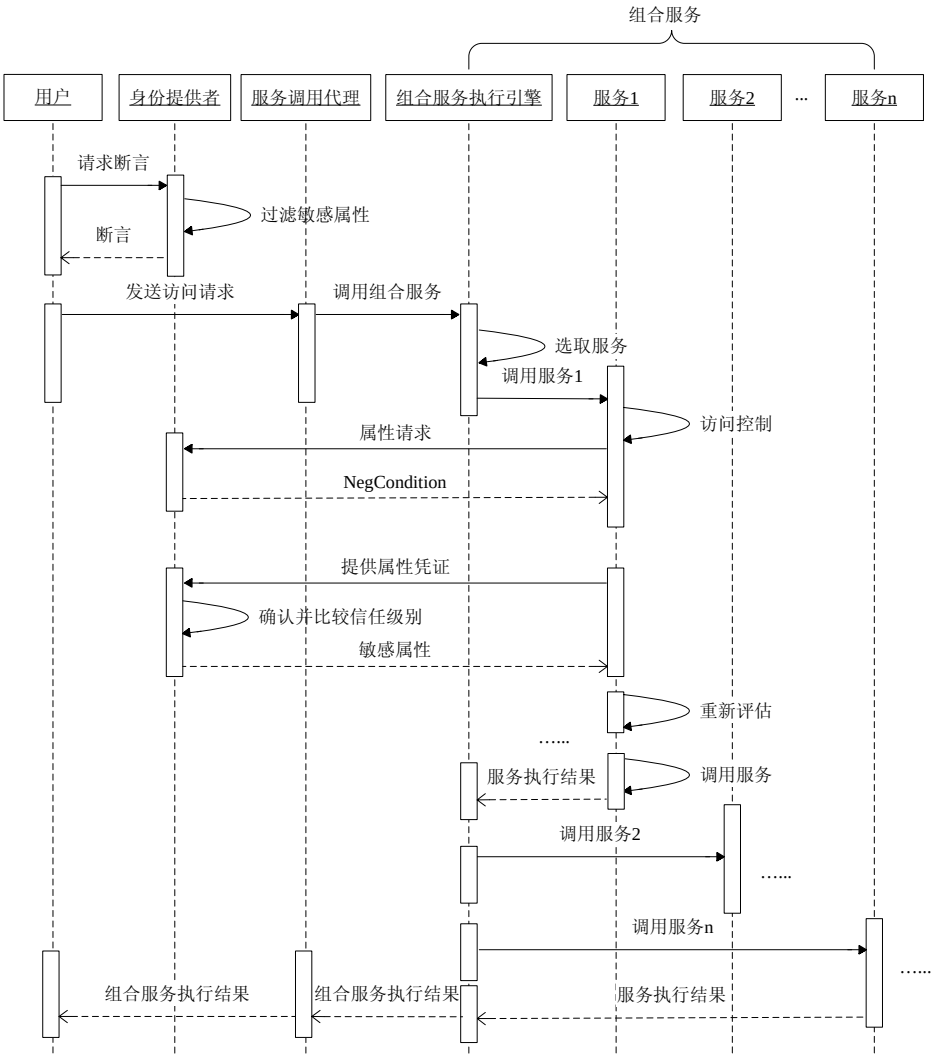

图 5-3 组合服务敏感属性保护流程

根据第四章，服务并不会直接拒绝请求，而是要协商。服务提供者会向用户发送协商建议 neg-sug=<negId,GRQS,{<SubAttrNotEnough,title>}>。如果用户接受协商，修改其策略为 p=<doctor zhang,3.0,{<departmentclass,3.0,Φ>,<title,3.5,{<departmentclass = 区政府,3.0>}>}> 。用户返回协商响应 neg-response=<negId,GRQS,{<SubAttrNotEnough,<title,true>>}>。GRQS 向 A 医院身份管理系统请求属性。A 医院身份管理系统在计算信任级别后，要求 GRQS 提供单位类别 departmentclass 属性。当服务 GRQS 提供其区政府属性凭证后，A 医院身份管理系统需对凭证进行验证，并再次比较信任级别，如满足条件，才返回张医生的 title 属性。评估成功后，调用服务 GRQS。其它服务的处理与上述方法类似。整个处理过程的序列图如图 5-4 所示。

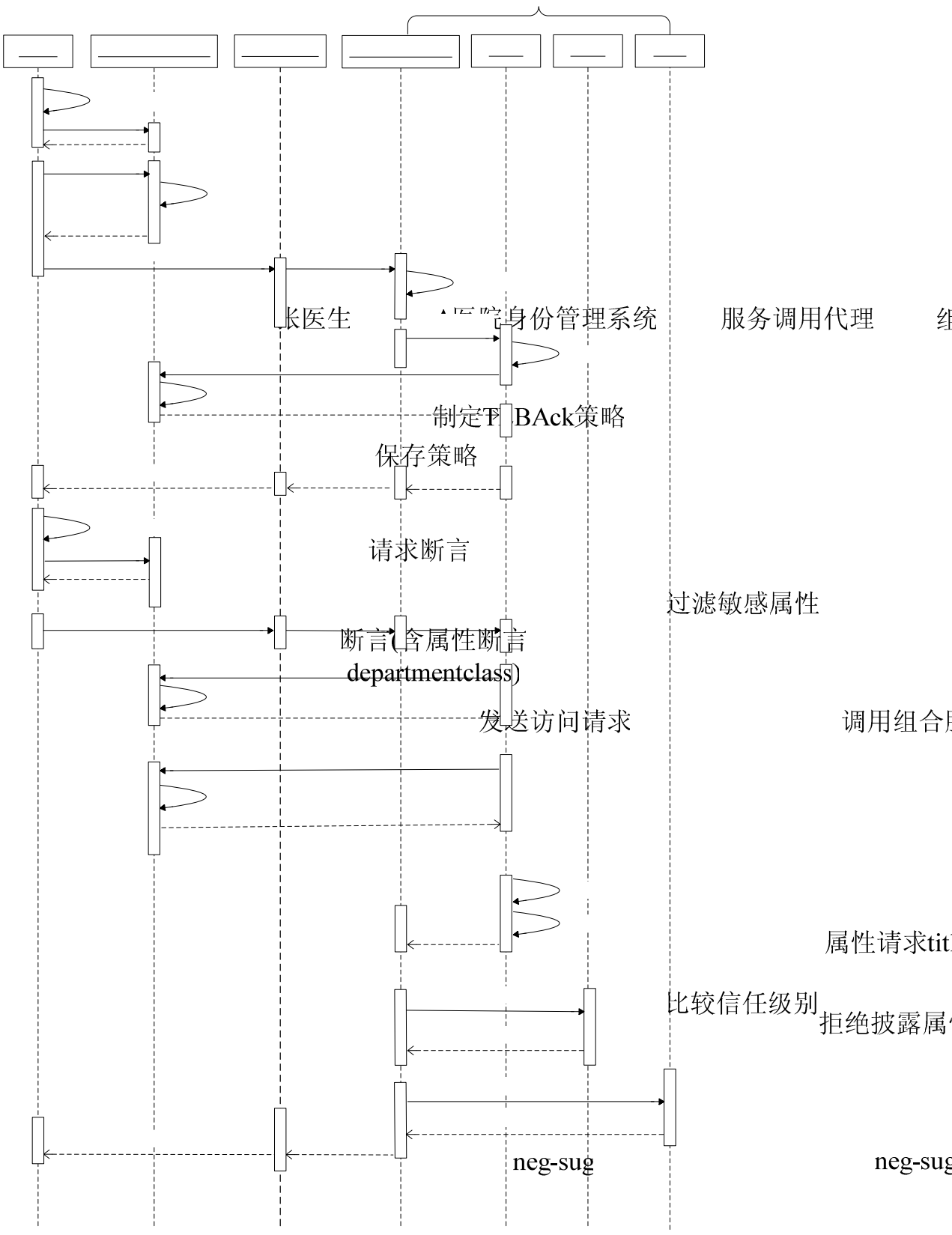

图 5-4 敏感属性保护示例

5.4 模型安全性分析

在第四章中为了增强访问控制的自适应性，引入了协商机制，可以通过协商来获得用户的属性。但用户的属性有的是敏感的，如果每个服务提供者都可以获得用户的任何属性，那么必然会导致敏感属性的泄漏。

本章提出了基于信任级别和协商机制的敏感属性保护方法，定义了一个TLBAck 策略，包括属性的信任级别及协商条件。可以从以下两个方面对敏感属性进行保护：

1. 敏感属性过滤

用户在向身份提供者请求断言时，身份提供者会根据用户制定的 TLBAck 策略进行敏感属性过滤，只有当属性的信任级别小于服务信任级别时，才加入到属性断言陈述中。规则 5- 1 敏感属性过滤和 5.2.4 中的算法 1 都分别描述了敏感属性过滤过程。这个过程保证了敏感属性不被加入到断言中，从而防止了敏感属性泄漏。如 5.3.2 中的应用示例，A 医院身份管理系统在构造断言时，根据属性级别和 ServiceTLConstraint 比较，只提供了 departmentclass 属性，而不提供 title 属性。

2. 属性查询

服务提供者在访问控制评估过程中，当需要某属性时，会向身份提供者请求属性，见 5.2.4 中的算法 2 和算法 3。算法 2 会把服务的信任级别和属性的信任级别进行比较，只有当服务的信任级别大于属性的信任级别时，才披露敏感属性。同时为了提高敏感属性保护模型的自治性，对于特别的情况，加了一些协商条件。算法 3 是敏感属性协商的过程，身份提供者向服务提供者透露 NegCondition，服务提供者返回属性凭证，经身份提供者验证后，再次对信任级别进行比较。只有满足属性条件且信任级别符合条件的服务提供者，才可以访问到敏感属性，实现了对属性的细粒度保护。如 5.3.2 中的应用示例，服务 GRQS 需获得用户的 title 属性，但其信任级别不够。经过协商，GRQS 只有提供其区政府的属性凭证，并经过再次的信任级别比较，才可以访问用户的 title 属性。

综上所述：该模型保护了用户的敏感属性，只有符合信任级别条件或者满足协商条件的服务提供者才可以访问敏感属性。

5.5 本章小结

针对目前用户敏感属性保护中的策略制定复杂，以及自治性差的不足，本章提出了一种基于 TLBAck 策略和协商机制的 Web 服务环境下的敏感属性保护模型。给出了模型的总体结构、形式化描述及相关算法。对组合服务的敏感属性保

护进行了研究，给出了示例说明。最后对模型的安全性进行了分析。该模型较好的解决了在访问控制过程中的用户敏感属性泄漏问题，策略制定简单，而且具有协商功能。

第六章 可扩展的 Web 服务安全系统原型设计

本章描述了一个可扩展的 Web 服务安全系统原型 EWSSSystem（Extensive Web Services Security System）。它集成了 Web 服务安全的相关规范，如使用 XML Signature 和 XML Encryption 来对 SOAP 消息内容进行加密和签名，使用 WS-Security 来传送安全令牌 token 来进行身份验证。同时它使用了 SAML 来进行跨域的单点登录，使用 XACML 来进行访问控制，使用了 SSL/TLS 技术保证通道的安全,使用 PKI 来进行通信双方的互相认证。本章首先给出 EWSSSystem 的体系结构，然后基于开源项目对系统中的单点登录、访问控制、敏感属性保护模块进行了详细的设计。

6.1 EWSSSystem 的体系结构

6.1.1 总体结构

如图 6-1 所示，EWSSSystem 结合了第三章的单点登录模型，第四章的访问控制模型，第五章的敏感属性保护模型。系统中各模块如下：

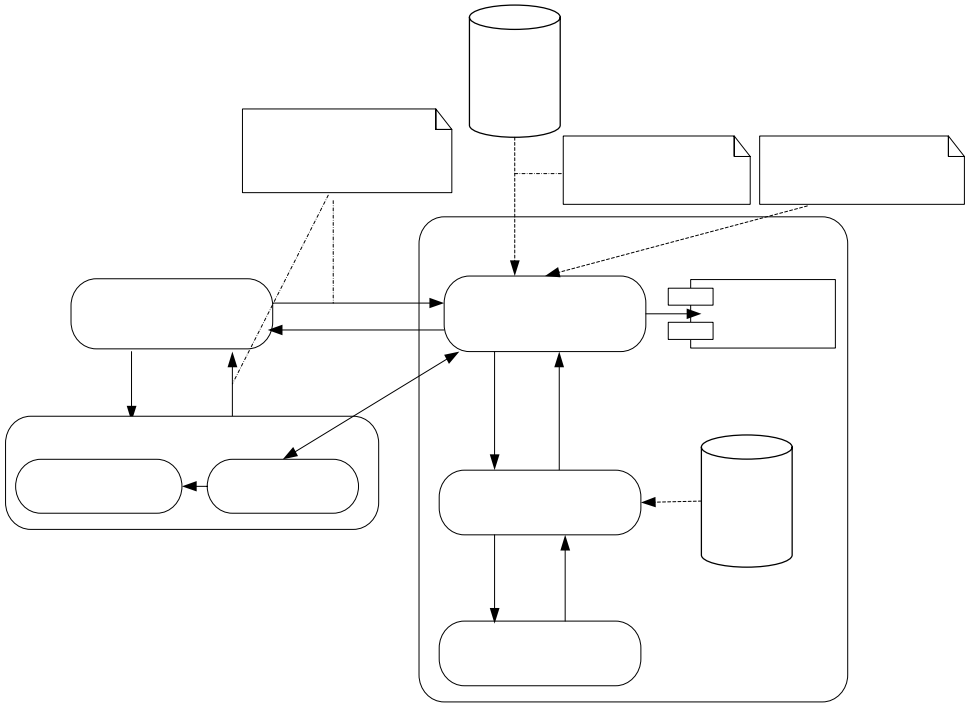

图 6-1 EWSSSystem 实现结构图

1. SOAP Client 为服务请求的发起者，为客户端程序。
2. Application System 模块充当身份提供者角色，当用户被验证为合法用户时，就发放相应的断言。
3. MyAttributeProtect 模块的功能是对用户的敏感属性进行保护。
4. AssertionResponse 模块供 SAMLReceiver 模块查询属性断言和验证断言。
5. SAMLReceiver 模块接收服务请求，对断言进行验证，并调用其它模块进行访问控制，同时可以取得主体、环境、资源属性。
6. MyPDP 是对客户端的请求进行评估。
7. MyNegotiation 是访问控制的协商模块，负责访问控制的协商过程。

EWSSSystem 执行的序列图如图 6-2 所示，SOAP Client 先从 Application System 中取得断言，然后访问服务提供者。SAMLReceiver 负责接收请求，并传递给 MyPDP 进行属性评估。当评估结果为部分接受时，则进行协商。MyNegotiation 模块负责整个的协商过程，首先生成协商建议，返回到客户端。客户端调整请求后，返回协商响应，MyPDP 和 MyNegotiation 重新进行评估，再判断客户端是否可以访问服务。MyAttributeProtect 负责整个单点登录和访问控制过程中的敏感属性保护，在单点登录构造断言时进行敏感属性过滤，在服务提供者向身份提供者进行属性查询时进行敏感属性保护并协商。系统中对于执行时间较长的操作均使用了异步调用，比如请求断言、请求评估、发送访问请求等。

6.1.2 层级结构

随着基于 XML 的协议的发展，如 WSDL,UDDI,SOAP 等，每一个协议都考虑了与其底层及上层的协议的兼容性。因为因特网本身是一个层级结构，所以也用层级结构来描述 Web 服务环境。如图 6-3 所示, 给出了 EWSSSystem 的层级结构图。

因为基于 SOAP 的 Web 服务的数据流机制已经广泛的被支持，所以 EWSSSystem 中也是以 SOAP 做为消息层的。在 SOAP 的下一层是网络层，SOAP 可以和任何种类的网络协议及 RPC 兼容。因为 HTTP 是最常使用的协议，所以 EWSSSystem 中以 HTTP 做为网络层协议，来传输 SOAP 消息。

在 SOAP 的上层，因为 Web 服务环境中的大部分协议都是基于 XML 的，所以 Xml Signature 和 Xml Encryption 在 EWSSSystem 中起着非常重要的角色。所以它们被定义为安全层协议。而 Web Service security 协议是保证可信的和安全的传输消息，所以被定义为传输层协议。

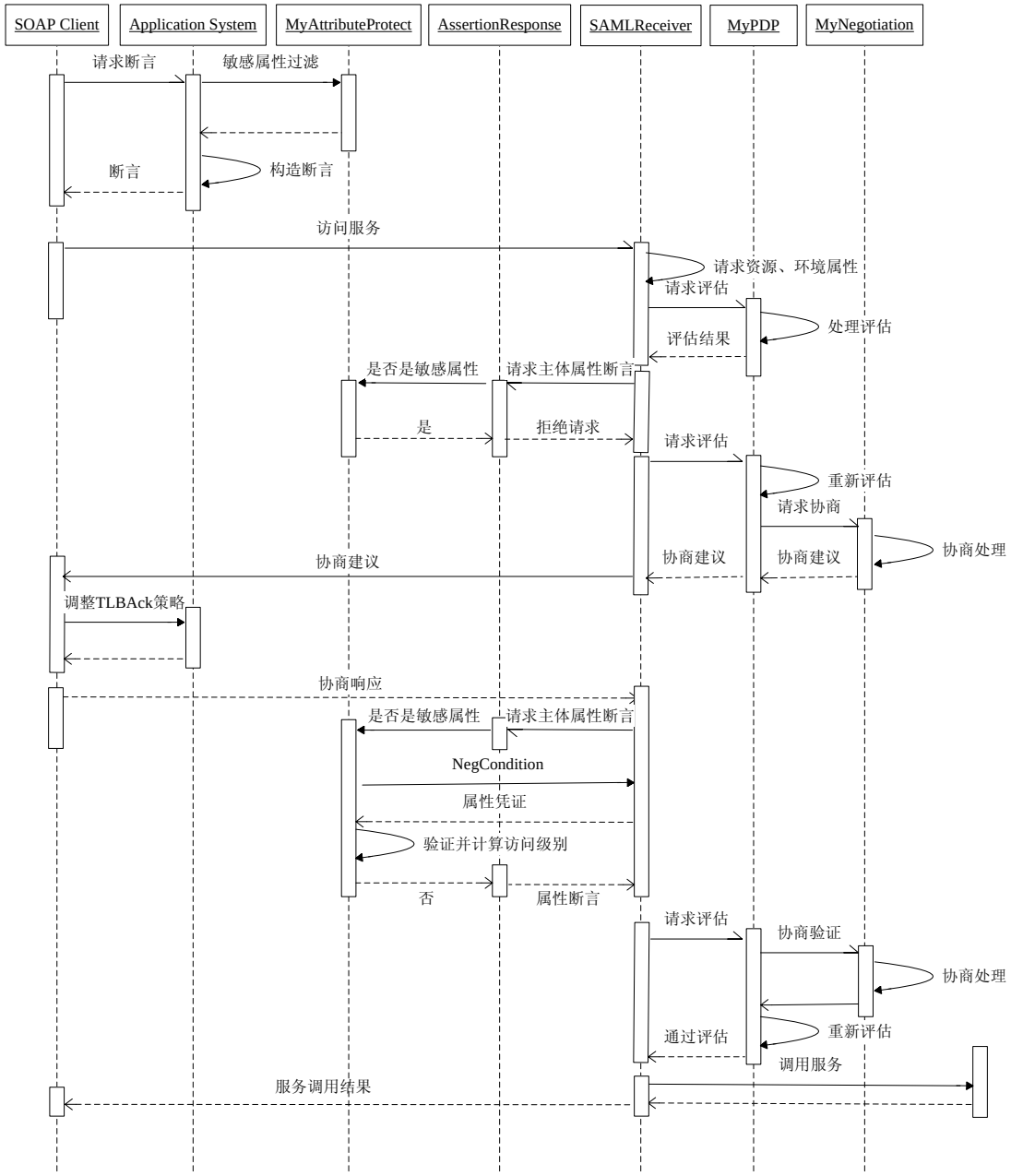

图 6-2 EWSSSystem 执行序列图

SAML 协议是用来验证用户身份的，被定义为验证层的协议。应用层协议是系统本身定义的一些协议，与具体的业务相关，如访问控制、协商协议等。

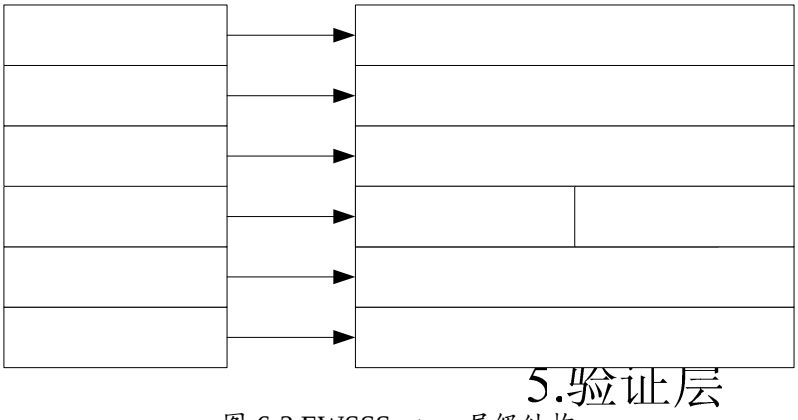

图 6-3 EWSSSystem 层级结构

6.1.3　结点栈结构

本节主要讨论 Web 服务环境下，一个结点怎样把包传送到另外一个结点，以及结点是如何处理这些包的。这些结点可能是一个 PC、服务器、分布式系统、客户端应用程序或 Web 服务提供者。如图 6-4 所示给出了结点的栈结构。与 OSI 的七层协议栈类似，Web 服务环境的栈结构在处理消息时，也是每一层用自己的标准去封装消息，对应的接收方用同样的标准去解开消息，而访问控制 XACML 作为结点的内部协议，结点与结点之间没有关联。

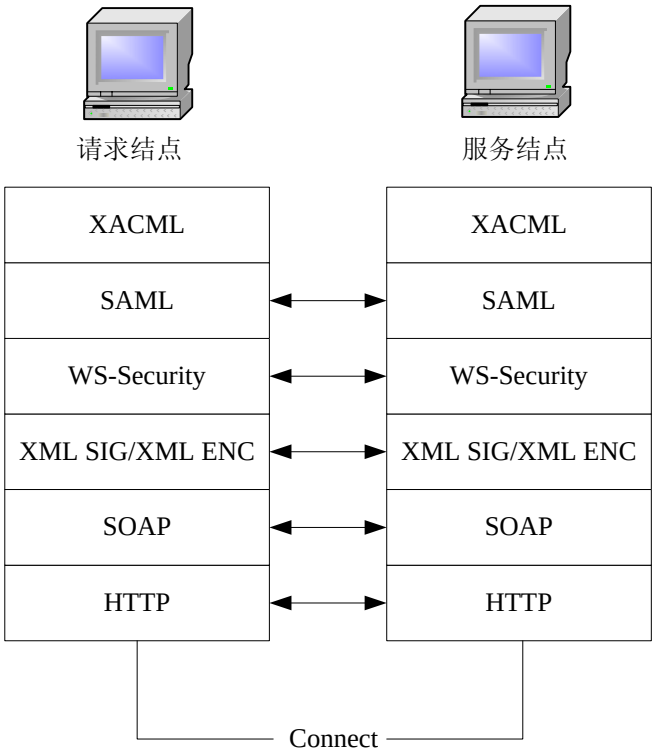

图 6-4 EWSSSystem 结点栈结构

6.2 主要相关技术

6.2.1 Web 服务的异步调用

随着 Web 服务技术的发展，越来越多的应用都集成了 Web 服务。但 Web 服务在执行时,时间的花销比较大。同步调用 Web 服务，请求和响应都在同一个线程中处理，客户端程序阻塞等待，当时间花销大时，用户体验较差，同时会可能产生超时错误。而异步调用中请求和响应不在同一线程，客户端无须阻塞，可以继续执行。所以对于客户端应用来说,使用异步调用 Web 服务是一种较明智的选择。

Apache Software Foundation 推出的开放源代码 Web 服务引擎 Apache Axis2[60] 为异步 Web 服务调用和实现提供了一流的支持。Axis2 具有两种在客户端提供异步性的机制和一种在服务器端提供异步性的机制[61]。

客户端异步允许在两个不同的线程中处理请求和响应消息。可以采用两种方法在客户端实现这种异步：1）非阻塞 API，客户端在调用 Web 服务后，继续执行其它工作，提供了一个回调对象来接收结果；2）传输级别的异步，提供两个通道，一个用来发送请求，一个用来接收。

服务器端异步采用一个消息接收器来接收 SOAP 消息，并同时启动另外一个线程来处理这个 SOAP 消息，并向客户端发送响应。这对于处理计算量较大的任务，有效防止了请求超时。

本系统中用到的 SOAP 消息，SAML 断言，以及 XACML 策略等都是基于 XML 的，在运行时都是非常耗时的，所以本系统在客户端和服务端都采用异步调用模式，缓解了性能问题，提高了用户体验。

6.2.2 开源项目

1. OpenSAML1.1API

OpenSAML1.1API[62]是一组开源的 Java 与 C++类库,它实现了 SAML（Security Assertion Markup Language）1.0 与 1.1 规范, EWSSSystem 使用了它的 Java 类库，来生成 SAML 断言、助诊、请求及响应消息。

2. WSS4J1.5.3

WSS4J（Web Services Security for JAVA）[63]是 OASIS 的 Web 服务安全规范 WS-Security 的一个实现。WSS4J 主要用来在 SOAP 消息头中添加安全信息来签名和确认 SOAP 消息。EWSSSystem 使用 WSS4J1.5.3 版本来对 SOAP 消息进行加密和签名。

3. SunXACML1.2

SunXACML[64]是一个由 Sun 公司发起的开源的工程，完全支持 OASIS 提出的 XACML 规范。SunXACML1.2 支持 XACML1.0 和 1.1 规范，EWSSSystem 基于它进行访问控制的实现。

6.3 EWSSSystem 的关键组件

EWSSSystem 主要有三个大模块：单点登录、访问控制与协商、用户敏感属性保护模块。这三个模块是紧密联系的，单点登录模块为访问控制和协商模块提供了属性断言陈述，以供其进行属性评估。用户敏感属性保护模块确保了在访问控制与协商过程中，身份提供者不会泄漏用户的敏感属性，只有满足一定信任级别和属性条件的服务提供者才可以访问用户的敏感属性。以下将对这三个模块进行详细的设计。

6.3.1 单点登录模块

单点登录模块主要有三个子模块，分别为：身份提供者对用户的身份认证、服务提供者对用户的身份认证、身份提供者的断言响应，最后给出了整个单点登录模块的执行序列图。

1. 身份提供者对用户的身份认证

类 IdAuthenticateImpl 主要完成身份提供者对用户的身份认证，并发放断言或助诊文件，并对它们的维护等，如图 6-5 所示，主要包括以下几个方法：

1) login()方法用来对用户的身份进行验证，用户提供自己的身份信息，可以是传统的用户名密码，也可以是数字证书类型。身份提供者根据本地数据库中的信息来对用户的身份信息进行验证,若合法，则建立 Session。

2) makeSAMLAssertion()方法用来创建新的断言，包括验证断言和属性断言两种类型，在生成断言过程中会调用接口 SensitiveAttrProtect 来进行敏感属性过滤。

3) makeSAMLArtifact()方法用来创建断言对应的助诊文件。

4) signSAML ()方法是对断言或助诊进行签名。

5) deleteSAMLAssertion()方法删除存储在系统中的断言。

6) deleteSAMLArtifact()方法删除助诊与断言的映射关系，助诊文件只能使用一次，使用后即被删除。

7) logout()方法是指用户退出登录，结束 Session，服务提供者方将无法根据助诊文件从身份提供者方获得断言。

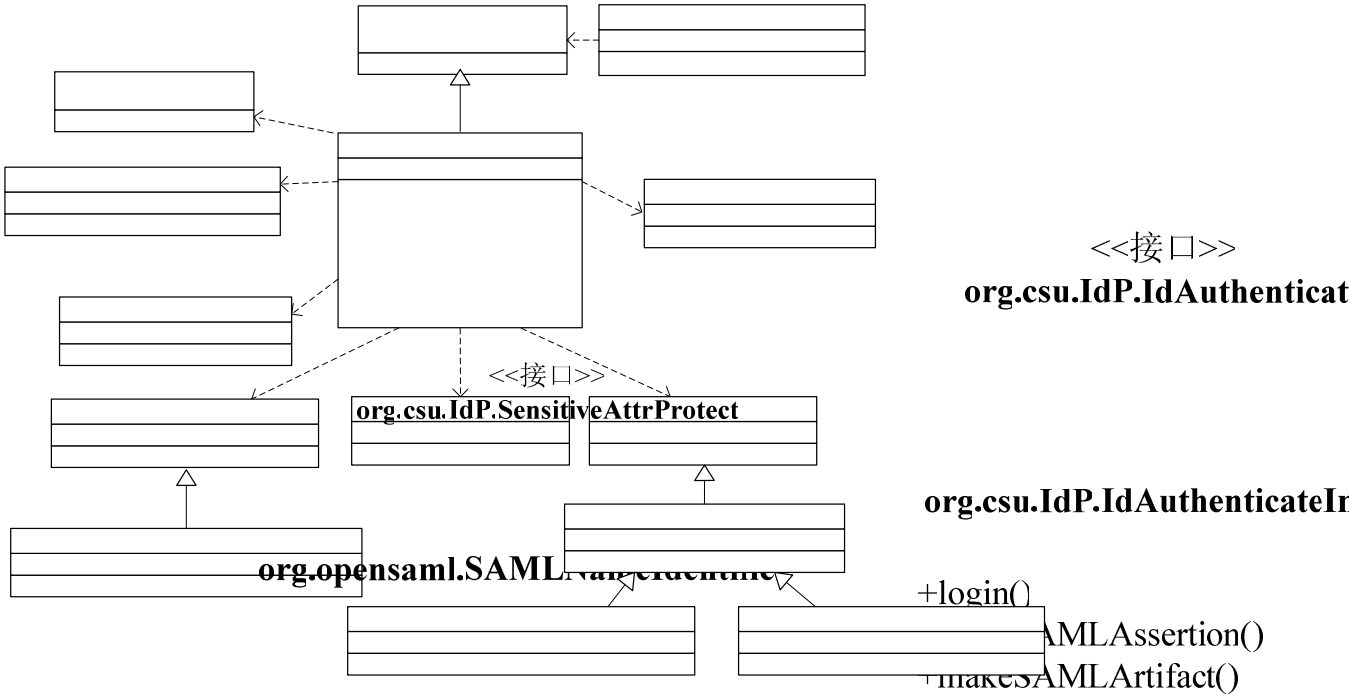

图 6-5 身份提供者对用户的身份认证模块

2. 服务提供者对用户的身份认证

类 SAMLReceiver 主要功能是在服务提供者方对客户端身份的验证，或者是向服务提供者请求断言。如图 6-6 所示包括如下方法：

1) login()方法主要完成服务提供者的身份验证功能，会调用下面的一系列的方法，若合法，则建立 Session。

2) receiveSAML()方法是从 SOAP 消息头中获取 SAML 消息，可能是助诊，也可能是断言，如果是助诊文件，首先调用 authenticateArtifact()方法确认其有效性然后调用 requestAssertionByArtifact()方法请求断言，如果是断言，就调用 authenticateAssertion()方法。

3) verifySignature()方法对助诊或断言的数字签名的确认。

4) authenticateAssertion()方法是对断言的有效性(如时间等)进行验证，如果通过，就会调用访问控制方法 accessControl()。

5) authenticateArtifact()方法是对助诊文件的有效性(如时间等)进行验证。

6) requestAssertionByArtifact()方法是服务提供者用它获取的助诊文件来向身份提供者请求断言。

7) accessControl()方法是服务提供者对用户的身份验证后，做访问控制评估，是否授予访问资源的权限。

8) logout()方法功能是用户退出登录，结束 Session。

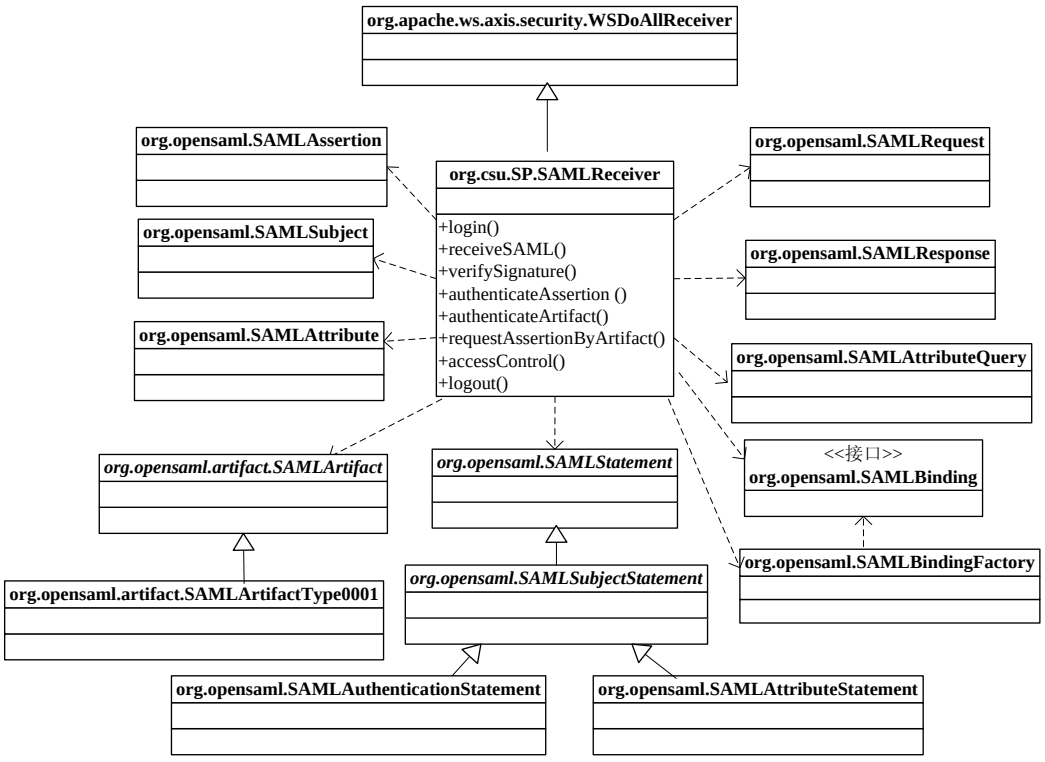

图 6-6 服务提供者对用户的身份验证模块

3. 身份提供者方的断言响应

类 AssertionRespServlet 是一个 Servlet 类，接收身份提供者的 HTTP 请求（请求中包含的是一个 SOAP 消息），调用类 AssertionResponder 进行请求的处理，然后返回响应。类 AssertionResponder 主要是用来对服务提供者的请求的响应，服务提供者发送了 SAML 请求，请求中包含了助诊文件或属性查询。身份提供者查询相应的断言并给予响应。如图 6-7 所示，具体的方法如下：

1) receiveSAML()方法是从 SOAP 消息中提取 SAMLRequest 消息，供其它方法处理，这个请求消息中既可能是一个助诊文件，也可能是一个属性查询文件；

2) verifySignature()方法是确认助诊文件的签名。

3) authenticateArtifact ()方法是对助诊文件的有效性进行验证。

4) retrivalAssertion()方法主要是根据助诊文件来检索对应的断言文件。

5) respondArtifact()方法主要是返回助诊文件对应的断言文件。

6) respondAttribute()方法是对服务提供者的属性查询给予响应。

7) protectSensitiveAttr()方法是对敏感属性进行保护，在做属性响应之前，先要比较属性和服务的信任级别，如果不满足，还需要进一步的协商，调用接口 SensitiveAttrProtect。见 5.2.4 节算法 2 和算法 3。

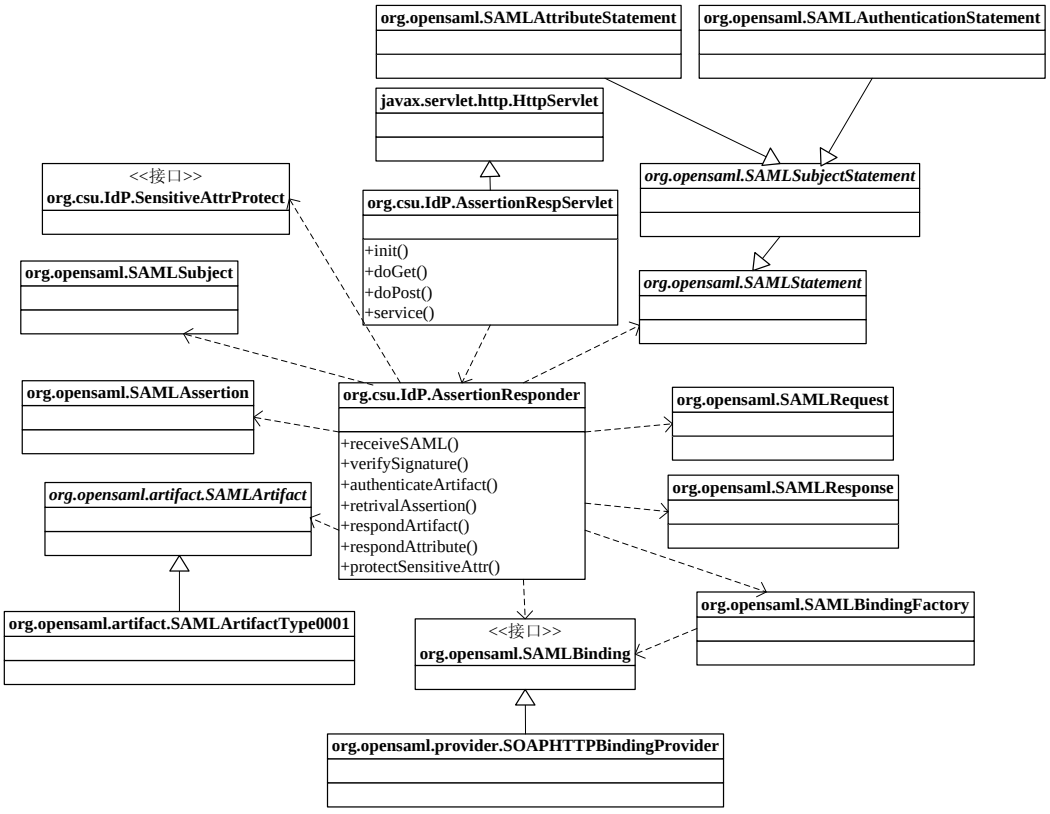

图 6-7 身份提供者方断言响应模块

6.3.2 访问控制与协商模块

1. XACML 格式的请求生成

策略的定义以 XACML 策略定义，主体、环境、资源属性及参数在策略定义中都以属性存在。

访问请求中的属性信息是以 SAML 格式的断言存在的，服务提供方须解析 SOAP 消息，把 SOAP 消息头中的断言提取出来，并在 SOAP 消息体中取出访问请求的服务参数，构成 XACML 访问请求，供评估引擎评估。

如图 6-8 所示，类 SAMLReceiver 是上述单点登录模块中的服务提供者身份认证主类，它解析含有断言的服务访问请求 SOAP 消息，并调用接口 XACMLRequestBuilder，构造 XACML 访问请求。接口 XACMLRequestBuilder 的实现类 XACMLRequestBuilderImpl 主要包含以下方法：

1) createSubjectAttr()方法功能是创建主体属性，从 SAML 断言中提取属性断言，构造成主体断言；因为服务参数也是主体提供的，所以也看做主体属性。

2) createResourceAttr()方法是创建资源属性。

3) createEnviromentAttr()方法是创建环境属性。

4) createActionAttr()方法是把服务请求调用的方法也做为一个属性。

5) createRequestCtx()方法是把根据上面建立的属性来构造请求上下文，然后传递给 PDP 模块进行评估。

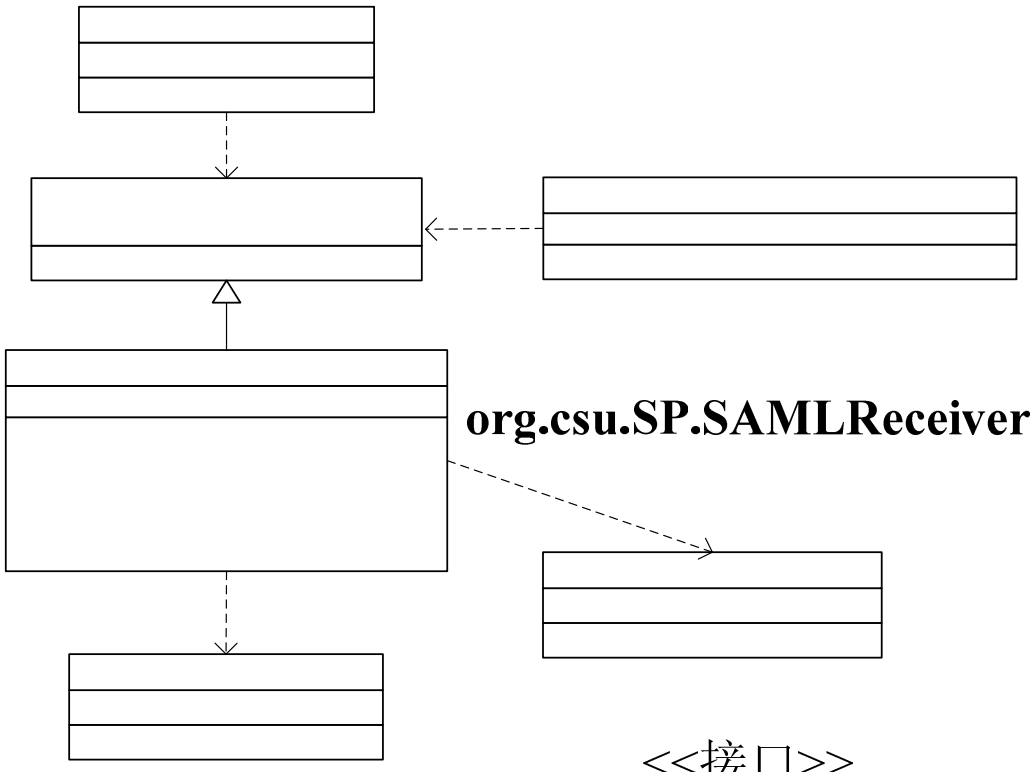

图 6-8 XACML 访问请求生成模块

org.csu.SP.XACMLRequestBuilder

2. 评估与协商

如图 6-9 所示，类 MyPDPImpl 是评估主类，负责接收访问请求，并产生响应结果。它有如下几个方法：

1) MyPDPImpl()构造函数。

2) evaluate()方法是对访问请求进行评估的主方法，调用类 PDP 进行评估。

org.csu.SP.XACMLRequestBuilderImpl

3) isNeedNegotiate()方法获得请求是否需要协商，如果评估的结果是允许，则无需协商，其它情况则查找是否有协商建议产生。

4) getNegoSug()获得协商建议，返回给类 SAMLReceiver。

+createSubjectAttr()
+createResourceAttr()
+createEnviromentAttr()
+createActionAttr()
+createRequestCtx()

类 EvaluateCondition 继承了类 Apply，根据策略对访问请求进行评估，主要有如下方法：

1) evaluate()是对访问请求与策略进行评估，算法如 4.2.4 的算法 2 评估属性。

2) requestAA()是向身份提供者的属性权威查询用户属性，以供评估需要。

3) provideAttributeCredential()方法是向身份提供者方提供服务提供者的属性凭证。

4) makeNegoSug()是生成协商建议，算法如 4.2.4 的算法 2 协商建议生成。

com.sun.xacml.ctx.Attribute

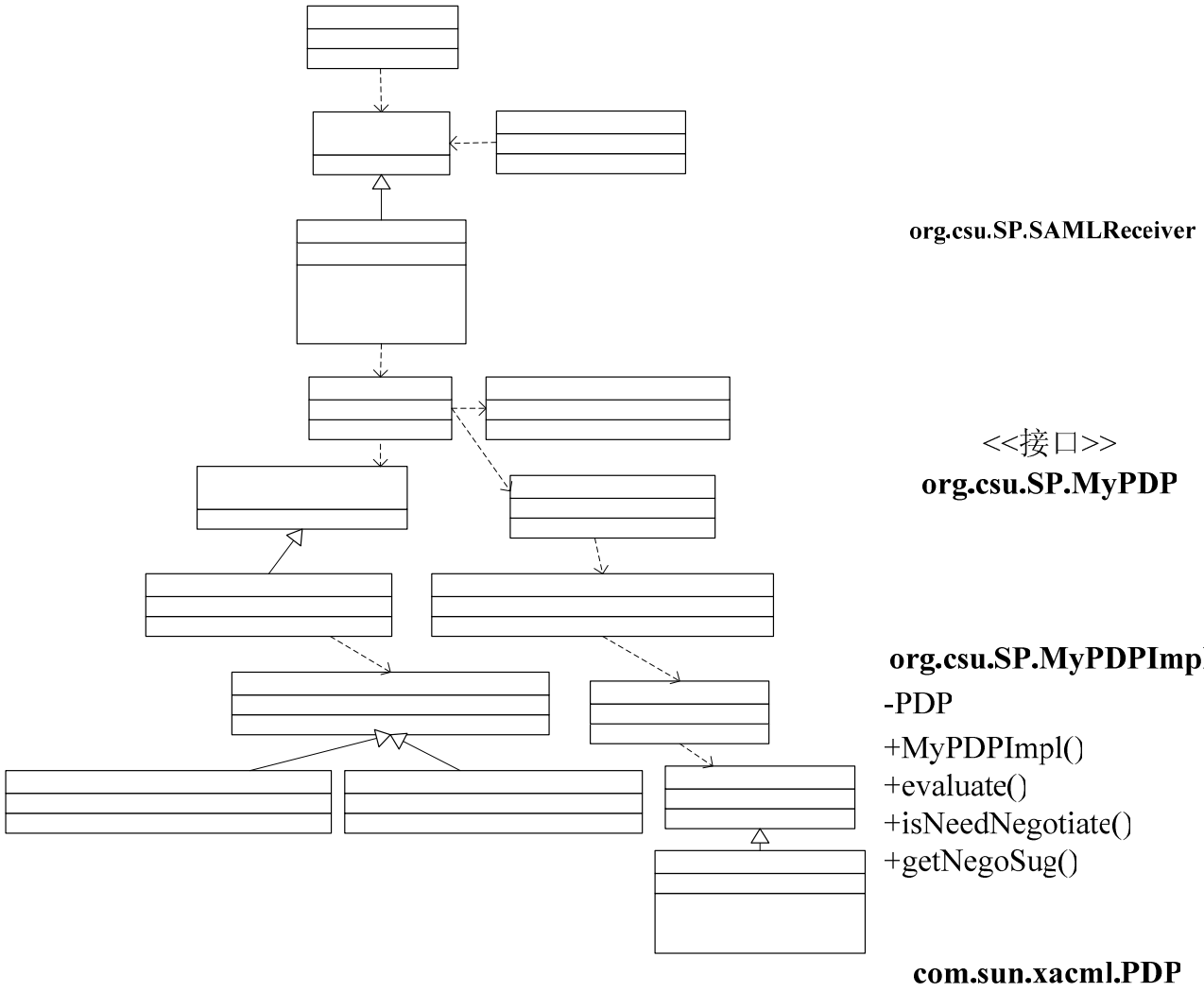

图 6-9 评估与协商模块

3. 执行协商

PDP 在做出协商决定后，将由类 NegoManagerImpl 来管理具体的协商过程，算法如 4.2.4 的算法 3 协商处理。如图 6- 10 所示，类 NegoManagerImpl 主要有如下方法：

1) negIdMatch()方法查询协商 Id 是否合法。
2) parameterMatch()方法是判断参数是否匹配。
3) getAttribute()方法是获取用户属性。
4) reEvaluate()方法是重新评估访问请求。
5) sendNegoSug()方法是发送协商建议。

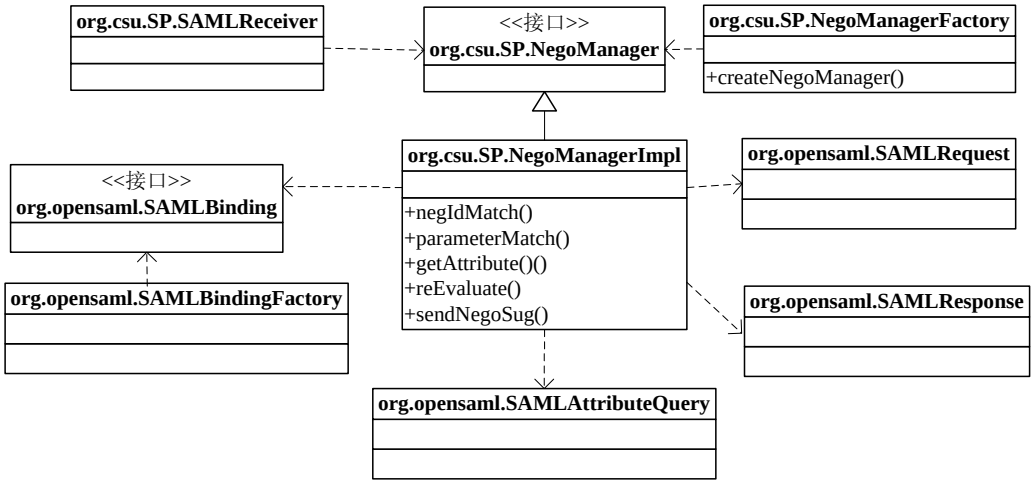

图 6- 10 协商执行模块

6.3.3 用户敏感属性保护模块

如图 6-11 所示，敏感属性保护接口 SensitiveAttrProtect，主要用在两个方面：1)类 IdAuthenticateImpl 在生成断言时，使用它来进行敏感属性过滤，实现算法见 5.2.4 中的算法 1；2)类 AssertionResponder 在处理属性断言查询时，使用它来进行敏感属性保护，包括比较信任级别和敏感属性协商，实现算法见 5.2.4 中的算法 2 和算法 3。

类 TLBAckPolicyEntity、TLBAckPolicyAttrEntity、TLBAckAttrNegCondEntity 是一些实体类，分别用来表示一个 TLBAck 策略、策略中的属性、属性的协商条件。这三个实体持久化在数据库中的。类 SensitiveAttrProtectImpl 实现了接口 SensitiveAttrProtect，主要有如下方法：

1) filterSensitiveAttribute()方法的功能是在身份提供者构造断言时，进行敏感属性的过滤。

2) getNegCondition()方法是向协商的另一方泄漏协商条件。

3) isConditionTrue()方法判断服务提供者提供的属性凭证是否满足协商条件。

4) compareTL()方法是用来比较属性和服务提供者的信任级别，判断该服务是否能访问对应的属性。

5) addTLBAckPolicy()方法是添加一个 TLBAck 策略。

6) editTLBAckPolicy()方法是修改一个 TLBAck 策略。

7) deleteTLBAckPolicy()方法是删除一个 TLBAck 策略。

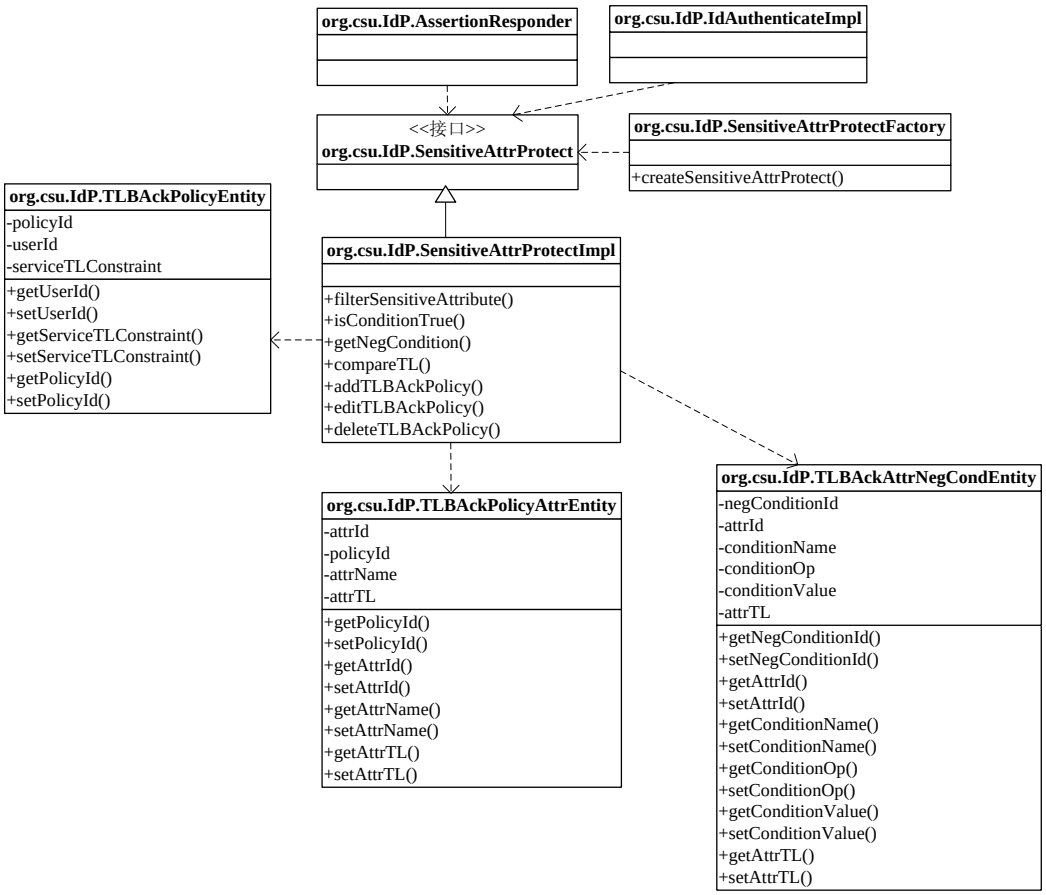

图 6-11 用户敏感属性保护模块

6.4 本章小结

基于开源项目 OpenSAML1.1API、WSS4J1.5.3 、SunXACML1.2，设计了 Web 服务环境下的单点登录和访问控制原型 EWSSSystem,并详细设计了单点登录、访问控制与协商、用户敏感属性保护等模块。EWSSSystem 可以适应 Web 服务环境下的单点登录和访问控制，并对敏感属性进行了保护。

第七章 总结与展望

7.1 本文总结

Web 服务环境下的新特性使得传统的单点登录与访问控制方法已不再适用，所以本文提出了可以适应 Web 服务环境的单点登录与访问控制模型，并给出了敏感属性的保护方法。

本文的工作主要有以下几点：

1.提出了一种基于 SAML 的 Web 服务单点登录模型

基于 SAML 标准，本文设计了 Web 服务单点登录模型，给出了单点登录的 Push 和 Pull 模式的实现方法，并基于 Push 模式实现了组合服务的单点登录，解决了当前的一些跨域单点登录方法耦合性强和互操作困难的问题。对模型的安全性做了分析。模型的安全性较好，可以适用于 Web 服务环境的单点登录。

2.提出了一种具有协商机制的基于属性的 Web 服务访问控制模型

给出了一种基于用户、环境、资源属性的 Nego-ABAC 模型，解决了 Web 服务环境下由于对用户身份不可知而引起的访问控制困难以及由于业务快速变化导致角色扩散等一系列问题。同时引入了协商机制，服务提供者与用户可以就访问请求中的服务参数和用户属性进行协商，使得访问控制具有自治性和自适应性。并给出了基于 XACML 的 Nego-ABAC 实现结构。对 Nego-ABAC 模型的安全性进行了分析。Nego-ABAC 模型安全性较好，具有自适应能力，适合于 Web 服务环境。

3.提出了一种基于信任级别和协商机制的用户敏感属性保护模型

针对目前用户敏感属性保护中的策略制定复杂，以及自治性差的不足，给出了一种基于信任级别和协商机制的 Web 服务环境下的敏感属性保护模型。模型中提出了一种 TLBAck 策略，基于它来对敏感属性进行保护。首先在构造断言时对敏感属性进行过滤，其次在服务提供者请求属性时对敏感属性进行保护并可以通过协商来判定是否披露敏感属性。策略制定简单，而且具有协商功能。对组合服务的敏感属性保护做了详细的讨论。模型较好的解决了在访问控制过程中的敏感属性泄漏问题。

4.设计了一个可扩展的 Web 服务安全系统原型

基于开源项目 OpenSAML1.1API、WSS4J1.5.3 、SunXACML1.2，设计了可扩展的 Web 服务安全系统原型 EWSSSystem,并详细设计了单点登录、访问控制与协商、用户敏感属性保护等模块。EWSSSystem 可以适用于 Web 服务环境下

的单点登录和访问控制，并对敏感属性进行了保护。

7.2 进一步研究方向

1. 把单点登录和访问控制考虑到服务组合过程中去。本文考虑的组合服务，是组合好后，成员服务才对用户进行验证和授权。这种方法存在缺陷，因为用户可能无法通过成员服务的验证和授权，不能使用该成员服务。所以较好的策略是在动态服务组合时，成员服务就对用户就进行验证和授权，如果用户不满足成员服务的验证和授权条件，组合服务就重新选择成员服务，这样组合出来的服务利用率将会更好一些。

2. 协商过程的自动化，利用本体的思想来对属性进行描述，使得属性匹配起来更加容易。

3. 目前互联网上的服务的信任级别还没有一个通用的计算标准，下一步将会建立一个模型来对服务的信任级别进行计算，并扩展 UDDI 等服务注册中心，使每个服务的信任级别都可以被查询。

4. 对 EWSSSystem 进行原型实现，并应用到一些项目中。

参考文献

[1] W3C.Web Services Architecture.http://www.w3.org/TR/ws-arch/

[2] Papazoglou M, Georgakopoulous D. Service-Oriented Computing. Communication of the ACM, 2003, 46(10):25~28

[3] Papazoglou M, Traverso P, Dustdard S, et al. Service-Oriented Computing Research Roadmap. In: Dagstuhl Seminar Proceedings on Service Oriented Computing (SOC). 2006

[4] Mike P. Papazoglou,Willem-Jan van den Heuvel.Service Oriented Architectures:Approaches, Technologies and Research Issues. The VLDB Journal The International Journal on Very Large Data Bases,2007,16(3): 389-415

[5] LiangJie Zhang, Jia Zhang, Hong Cai. Services Computing. BeiJing:TsingHua University Press,2007

[6] 岳昆,王晓玲,周傲英. Web 服务核心支撑技术:研究综述.软件学报,2004,15(3):428~442

[7] Object Management Group. Common object request broker architecture: core specification, version 3.0.3.2004

[8] Microsoft.DCOMArchitecture. http://msdn.microsoft.com/ en-us/library/ ms-809311.aspx

[9] 哈特曼(Hartman,B.)等著.杨硕译.全面掌握 Web 服务安全性.北京:清华大学出版社,2004

[10] 奥尼尔(O'Neill,M.)等著.冉晓旻,郭文伟译.Web 服务安全技术与原理.北京:清华大学出版社,2003

[11] 林满山,郭荷清.单点登录技术的现状及发展.计算机应用,2004,24(6):248-250

[12] Microsoft. .Net passport review guide. http://www.microsoft.com/ net/services/passport/review_guide.mspx, 2003

[13] Liberty Alliance Project. Liberty Architecture Overview V1.1 . http:// www. projectliberty.org,2003

[14] IBM.Web Services Federation language(WS-Federation) Version1.0. http://www.ibm.com/developerworks/library/specification/ws-fed/

[15] OASIS.SAML.http://www.oasis-open.org/committees/tc_home.php?wg_abbrev= security

[16] OASIS. Assertions and Protocol for the OASIS Security Assertion Markup

Language(SAML).http://www.oasis-open.org/committees/download.php/2290/oasis-sstc-saml-1.0.zip

[17] OASIS. Bindings and Profiles for the OASIS Security Assertion Markup Language(SAML).http://www.oasis-open.org/committees/download.php/2290/oasis-sstc-saml-1.0.zip

[18] 韩伟,范植华.基于SAML的单点登陆技术在Web服务中的应用研究.计算机工程与设计, 2005,26(3): 634~636

[19] 胡剑,寇雅楠.基于SAML的Web服务中联合单点登录的设计与实现.制造业自动化, 2007, 29(10): 79~81

[20] 陈科、佘堃、黄迪明,基于安全断言标记语言辅件技术的单点登录系统分析,计算机应用, 2005, 25(11): 2574~2576

[21] 吴鹏,吉逸. 基于 SAML 的安全服务系统的设计.计算机应用研究,2004,21(11): 127~129

[22] Flavio O.Silva, Joao A, A.Pacheco, et al.A Web Service Authentication Control System Based on SRP and SAML.In:Proceedings of the IEEE International Conference on Web Services(ICWS'05).2005

[23] Roosdiana Wonohoesodo, Zahir Tari.Role-Based Access Control System for Web Services.In: Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04).Shanghai.2004.49~56

[24] R. Bhatti, E. Bertino, A. Ghafoor. A Trust-based Context-Aware Access Control Model for Web Services. In:Proceedings of the IEEE International Conference on Web Services (ICWS'04).California: Springer Netherlands,2004. 83~105

[25] X Feng, L GuoYuan, H Hao, et al. Role-based Access Control System for Web Services.In:Proceedings of the Fourth International Conference on Computer and Information Technology.2004.357~362

[26] S HaiBo, H Fan.A Context-Aware Role-Based Access Control Model for Web Services.In:Proceedings of the IEEE International Conference on e-Buiseness Engineering.2005. 220~223

[27] M Liu, H Guo, J Su. An attribute and role based access control model for web services.In: Proceedings of the Fourth International Conference on Machine Learning and Cybernetics. Guangzhou: 2005.1302~1306

[28] E Bertino, A C Squicciarni, D Mevi.A Fine-grained Access Control Model for Web Services.In: Proceedings of the IEEE International Conference on Services Computing.2004.33~40

[29] E Yuan, J Tong. Attribute Based Access Control (ABAC) for Web Services. In:Proceedings of the IEEE Conference on Web Services (ICWS'05) .Orlando Florida:2005

[30] S HaiBo, H. Fan. An Attribute-Based Access Control Model for Web Services. In: Proceedings of Parallel and Distributed Computing, Applications and Technologies. 2006. 74~79

[31] 沈海波，洪帆．基于属性的授权和访问控制研究．计算机应用，2007, 27(1):114~117

[32] 沈海波，洪帆. Web服务中结合XACML的基于属性的访问控制模型．计算机应用, 2005, 25(12):2765~2768

[33] 唐成华，胡昌振．基于AC的XACML访问控制模型的设计及实现．计算机应用研究, 2006, 23(10): 133~136

[34] 史毓达，沈海波．基于 XACML 的 Web 服务访问控制模型．计算机应用研究,2007,24(6):87~91

[35] 傅鹤岗,李竞．基于属性的 Web 服务访问控制模型.计算机科学, 2007,34(5):111~115

[36] 韩涛,郭荷清,高英,等.一个 Web 服务访问控制模型.计算机科学,2007,34(10):159~163

[37] Vipin Singh Mewar, Subhendu Aich ,Shamik Sural. Access Control Model for Web Service with Attribute Disclosure Restriction. In:Second International Conference on Availability,Reliablility and Security(ARES'07)

[38] M.Winslett, T.Yu, K.E.Seamons, et al. Negotiating Trust on the Web. IEEE Internet Computing, 2002, 6(6):30~37

[39] Kent E.Seamons, Marianne Winslett, Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation. In: Proceedings of the Symposium on Network and Distributed System Security(NDSS'01).2001

[40] Kent E.Seamons, Marianne Winslett, Ting Yu,et al.Protecting privacy during online trust negotiation. In: Proceedings of 2nd Workshop on Privacy Enhancing Technologies.Springer-Verlag: 2002

[41] William H. Winsborough, Ninghui Li. Towards practical automated trust negotiation. In: Proceedings of the Third International Workshop on Policies for Distributed Systems and Networks (Policy 2002).IEEE Computer Society Press:2002. 92~103

[42] William H. Winsborough, Kent E.Seamons,Vicki E.Jones. Automated trust

negotiation. In :DARPA Information Survivability Conference and Exposition.IEEE Press,2000.88~102

[43] Ting Yu,Xiaosong Ma,Marianne Winslett.Prunes:An efficient and complete strategy for trust negotiation over the internet. In: proceedings of the 7th ACM Conference on Computer and Communications Security(CCS-7).2000: 210~219

[44] Ting Yu,Marianne Winslett,Kent E.Seamons.Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. In:Proceedings of the Eight ACM Conference on Computer and Communications Security(CCS-8)

[45] 李建欣, 怀进鹏, 李先贤.自动信任协商研究.软件学报,2006,（1）: 124 ~133

[46] Winsborough W H, Seamons K E, Jone V E. Automated TrustNegotiation. In: Proc of the DARPA Information Survivability Conf. and Exposition. New York: IEEE Press, 2000. 88～102

[47] Winsborough W H, Jacobs J.Automated Trust Negotiation in Attribute-based Access Control In:Proc of the DARPA Information Survivability Conference and Exposition(DISCEX),Washington,D.C.2003

[48] XML-Signature Syntax and Processing. W3C. http://www.w3.org/ TR/2002/REC-xmldsig-core-20020212/

[49] XML Encryption Syntax and Processing. W3C.http://www.w3.org/TR/ 2002/REC-xmlenc-core-20021210/

[50] OASIS.Web Services Security. http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=wss

[51] OASIS.SAML V1.0. http://www.oasis-open.org/committees/ download.php/ 2290 /oasis-sstc-saml-1.0.zip

[52] OASIS.SAML V2.0. http://www.oasis-open.org/committees/download.php/ 22385/sstc-saml-core-errata-2.0-wd-04-diff.pdf

[53] OASIS.XACML V1.0. http://www.oasis-open.org/committees/download.php /2406/oasis-xacml-1.0.pdf

[54] OASIS.XACML V1.1. http://www.oasis-open.org/committees/xacml/repository /cs-xacml-specification-1.1.pdf

[55] OASIS.XACML V2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[56] Thomas GroB.Security Analysis of the SAML Single Sign-on Browser/Artifact Profile.In: Proceedings of the 19th Annual Computer Security Applications

Conference.2003. 298~307

[57] SM Hansen, J Skriver, HR Nielson. Using static analysis to validate the SAML single sign-on protocol.In: Proceedings of the 2005 workshop on Issues in the theory of security.2005. 27~40

[58] IBM Developer Works. Security in a Web Services World: A Proposed Architecture and Roadmap, April 2002

[59] Seamons KE, Winslett M, Yu T, et al. Protecting privacy during on-line trust negotiation. In: Proceedings of the 2nd Workshop on Privacy Enhancing Technologies. Springer-Verlag: 2003.129~143

[60] Apache.AXIS2.http://ws.apache.org/axis

[61] IBM. 使用 Axis2 开发异步 Web 服务. http://www.ibm.com/developer-works/cn/webservices/ws-axis2/index.html

[62] Internet2. OpenSaml. http://www.opensaml.org/

[63] Apache.WSS4J.http://ws.apache.org/wss4j/

[64] Sun. SunXACML 1.2.http://sunxacml.sourceforge.net/

致谢

本文得以顺利完成，首先应该感谢我的导师李建华教授。在硕士研究生学习阶段，李老师严谨的治学态度、渊博的学识、正直向上的人格魅力和高尚的品德深深地影响了我的价值观、人生观，让我受益终生。李老师有着广阔的胸怀，“我不赞成你，但我支持你”这句李开复形容他的导师时说的一句话，也同样可以用来形容我的导师。每次我有想法的时候，即使李老师不赞成，但仍然支持我去做。

另外，要感谢实验室的马华、徐海军、许甸、丁昭华师兄以及师姐们，他们在学习和生活上给了我很多帮助，让我受益匪浅。感谢同级的李勇军、曾慧琼、李桂林、胡国晴、李鹏、李琳、佘科华、王斌等同学，他们在我的学习和生活上都曾给予我许多热忱的帮助，令我难已忘怀。还要感谢师弟师妹们和项目组的同事，他们的支持让我的工作和生活更加愉快、充实。

最后，尤其要感谢我的父母为我的成长所付出的心血，感谢王萍萍对我的支持和关心。他们的关爱是我完成学业的基础，是我应对所有挑战的力量之源。

攻读学位期间主要的研究成果

1. 参加科研情况：

［1］湖南省电子公文传输系统（项目主要成员）；
［2］工作流产品研发（项目主要成员）；
［3］湖南省财政厅国库支付系统（项目参与者）；
［4］中南大学医学遗传国家重点实验室临床信息管理系统（独立设计与开发）；
［5］湖南大学数字图书馆门户网站（项目参与者）。

2. 发表论文情况：

[1] 张慧,李建华,马华.一种基于 SAML 的 Web 服务单点登录模型研究与实现.计算机系统应用.拟刊在 2008 年 7 月(编号:ba012)

[2] 张慧,李建华,许甸,徐海军.一种工作流运行时流程回退方法的研究与实现. 计算机工程与科学,2008,30(5):88~91

[3] 张慧,李建华,李勇军,曾慧琼.一种具有协商机制的 Web 服务访问控制模型. 拟投稿

[4] 张慧,李建华,李勇军,曾慧琼.Web 服务环境下用户敏感属性保护方法研究. 拟投稿