基于分段模型检测的云服务跨域认证协议的形式化分析与验证

陈红松[1] 王 钢[2] 傅忠传[3]
(北京科技大学计算机与通信工程学院 北京 100083)[1] (铁道警察学院 郑州 450053)[2]
(哈尔滨工业大学计算机学院 哈尔滨 150001)[3]

摘 要 针对多个云服务之间的跨域认证问题,提出一种基于SAML协议的云服务安全认证方案。阐明了该方案的关键技术机制,建立了云服务安全认证协议抽象模型;采用Casper和FDR软件的组合,通过模型检测法对云服务认证协议进行了形式化分析与验证;通过对安全认证协议进行分段模型检测,解决了安全协议形式化分析验证导致的状态空间爆炸问题。模型检测软件的实验结果验证了云服务跨域认证方案的有效性及安全性。

关键词 云服务,认证协议,形式化分析,模型检测
中图法分类号 TP301.2 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.4.028

Cloud Services Cross-domain Authentication Protocol Formal Analysis and Verification Based on Fragment Model Check

CHEN Hong-song[1] WANG Gang[2] FU Zhong-chuan[3]
(School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)[1]
(Railway Police College, Zhengzhou 450053, China)[2]
(School of Computer Science, Harbin Institute of Technology, Harbin 150001, China)[3]

Abstract Aiming at the cross-domain authentication problem in multi-cloud services, a cloud service security authentication scheme based on SAML (Security Assert Markup Language) protocol was proposed. The key techniques and mechanism were clarified, and the abstract model of cloud service security authentication protocol was built. Using the combination of Casper and FDR software, cloud service authentication protocol was formally analyzed and verified by model checking method. By dividing the protocol to do model checking separately, the state space explosion problem in security protocol formal analysis and verification was resolved. The experiment results from model checking software prove that the proposed cross-domain authentication scheme is effective and security.

Keywords Cloud services, Authentication protocol, Formal analysis, Model checking

1 云服务相关安全规范与模型检测

1.1 云服务及Web Services安全技术

云计算是将大量的计算、存储、网络资源聚合为资源池并统一管理调度,通过虚拟资源池向用户提供所需的IT服务[1]。RSA2010已将云计算安全列为当前研究的焦点问题[2]。云服务安全正成为云计算安全的研究重点[3]。《伯克利云计算白皮书》提出了云计算发展中的十大挑战及应对策略[4],提出采用多个云提供商保障云服务的可用性。

云服务通常是一种基于Web Services的服务。2006年2月15日,结构化信息标准推进组织(Organization for Advancement of Structured Information Standards, OASIS)宣布通过了WS-Security版本1.1作为OASIS标准[5]。

1.2 SAML与XKMS规范

2003年初,OASIS发布了安全断言标记语言(Security Assertion Markup Language, SAML)1.1版本,来自25家公司的55名专家参与了该规范的制定,SAML是多台服务器之间需要共享认证信息时使用的协议规范。SAML基于XML标准交换安全信息,支持单点登录(Single Sign On, SSO),用于在不同的安全域之间交换认证和授权信息。2005年,OASIS发布SAML 2.0[6]。SAML 2.0消除了以往多协议复杂性,成为当前通用的技术标准。XML密钥管理规范(XML Key Management Specification, XKMS)为访问公钥基础设施(Public Key Infrastructure, PKI)提供了一种规范方法。

1.3 云安全联盟

云安全联盟(Cloud Security Alliance, CSA)在发布的《云计算关键领域安全指南》中明确指出:为了能应用强认证,云应用程序都应该支持委托认证给使用此程序的企业,例如通过SAML服务。

到稿日期:2015-03-13 返修日期:2015-07-07 本文受北京市科技计划项目(D141100003414002),"十二五"国家863高技术研究发展计划重大专项:亿级并发云服务器研制(2013AA01A209),中央高校基本科研业务费项目(FRF-TP-14-042A2),北京市自然科学基金(4142034),北京市青年英才计划(YETP0380),国家留学基金管理委员会资助。

陈红松(1977-),男,博士,副教授,主要研究方向为云计算与云安全、物联网安全、软件安全,E-mail:chenhs@ustb.edu.cn;王 钢(1958-),男,教授,主要研究方向为网络信息安全;傅忠传(1970-),男,博士,副教授,主要研究方向为并行计算、系统模拟。

本文正是在这样的背景下，提出基于 SAML 协议的云服务安全认证方案，并对其安全性进行形式化分析，通过模型检测软件工具验证其安全性。

1.4 Web Services 的模型检测

丹麦 Aalborg University 的 A. P. Ravn 等在 2013 年采用 Uppaal 软件对 WS-BA 规范(Web Services Business Activity)产生的状态-迁移表进行检测分析[7]，发现按照 WS-BA 规范执行有可能到达某些非法状态，并对其提出了修改建议。

澳大利亚 Adelaide 大学的 Quan Z. Sheng 等根据 Web Service 的功能和行为将其分为控制行为和操作行为两类[8]，并采用状态图将这两类行为及交互过程进行模型化，通过设计阶段的分类建模有助于构建复杂的 Web Services 云服务系统。

澳大利亚 Innsbruck 大学的 Philipp Zech 提出了模型驱动的云计算安全测试方法[9]，其通过建立基于 UML 概念的系统模型得到误用测试用例，实现自动风险分析以及错误用例消减。

清华大学的林闯教授等提出了可管、可控、可度量云计算安全模型[10]，从云计算安全指标体系、指标的形式化描述、安全模型与分析方法等方面展开述论，并提出了一种以多队列多服务器模型为基础的云计算安全建模方法。

北京科技大学的陈红松副教授提出了基于多标签的大数据访问控制模型[11]，即访问控制服务模块根据云服务中数据访问模型的主体标签和客体标签自动分配访问权限，进行访问风险的自动识别与控制。

2 基于 SAML 协议的云服务认证模型

2.1 SAML 工作原理

在充分研究 Web Services 和 SAML、XKMS 等技术的基础上，提出如图 1 所示的云服务安全认证方案。

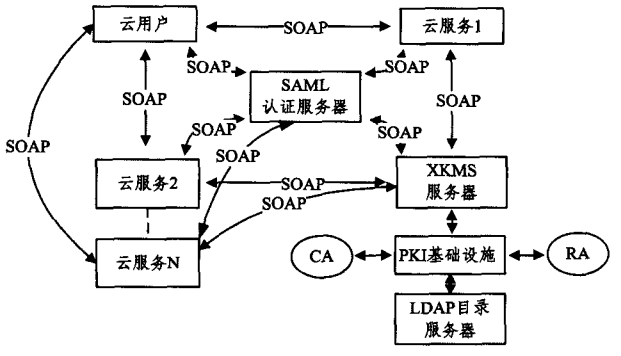

图 1 基于 Web Services 的云服务安全认证方案

本文提出的方案中包括了 5 种角色：云用户、云服务、SAML 认证服务器、XKMS 服务器和 PKI 基础设施。云用户是服务发起方；云服务是服务提供方；SAML 认证服务器对云用户进行认证，发放 SAML 属性断言，并响应云服务对云用户属性断言的查询；XKMS 服务器的功能是对 SAML 认证服务器和云服务发放证书，基于 XKMS 的管理层提供密钥管理服务，形成 SAML 认证服务器和多个云服务间的信任域；XKMS 服务器利用底层的 PKI 基础设施来保存和绑定密钥；通过 PKI 的 CA(Certificate Authority)实现对云用户、云服务提供者、SAML 认证服务器公钥证书的签发及管理，通过 PKI 的 RA(Register Authority)实现对公钥证书申请者的信息录入、审核，LDAP(Light Directory Access Protocol)目录服务器用于存储用户的证书和黑名单信息，用户可通过标准的 LDAP 协议查询自己或其他人的证书和下载黑名单信息。本文设计的云服务安全认证方案采用基于 SOAP 的安全传输技术，通过 XML 签名和 XML 加密实现端到端之间传输消息的安全保护。

云服务安全认证方案的执行流程为：云服务及 SAML 服务器首先到 XKMS 服务器进行公私钥注册，在 PKI 基础设施的支撑下获得身份许可，以此建立包含云服务和 SAML 服务器的信任域。具体步骤如下：

(1)云服务向 XKMS 服务器请求身份，基于 PKI 基础设施获取 XKMS 身份证明；

(2)SAML 认证服务器向 XKMS 服务器请求身份，基于 PKI 基础设施获取 XKMS 身份证明；

(3)SAML 认证服务器向各个云服务请求认证，获取云服务的信任，与云服务构造信任域；

(4)云用户向 SAML 认证服务器请求身份许可及属性断言；

(5)SAML 认证服务器对云用户进行认证并产生属性断言；

(6)云用户使用 SAML 属性断言向 SAML 信任域内的云服务发送服务请求；

(7)云服务通过用户的 SAML 属性断言进行认证，把认证结果返回给用户。

云服务安全认证协议的安全目标为：

(1)身份认证。SAML 认证服务器对云用户进行集中认证，通过 SAML 认证服务器实现云用户和云服务之间的双向认证。

(2)密钥分配。在云用户和云服务之间建立一个共享会话密钥并保证会话密钥的机密性。SAML 认证服务器作为可信任的第三方，为云用户和云服务生成并分发会话密钥。

(3)认证断言。云服务借助云用户的 SAML 认证属性断言实现多个云服务对一个云用户的跨域认证。

2.2 云服务安全认证协议抽象模型

云服务安全认证协议的抽象模型如图 2 所示。

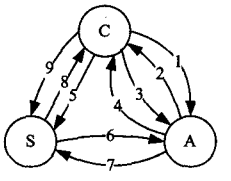

图 2 云服务安全认证协议的抽象模型

参与协议的 3 个主体分别为 C(云用户)，A(SAML 服务器)和 S(云服务)。在协议中，$Cert_a$、$Cert_c$、$Cert_s$ 分别为 3 个主体的公钥证书，这个证书由 XKMS 服务器发放，内容包括在协议中被使用的 3 个主体各自的公钥，私钥仅由主体自己保存。协议中主体可以通过 XKMS 服务器的公钥来互相验证证书，并获得对方公钥。$Token_c$ 为 SAML 服务器向云用户 C 发放的属性断言，用仅有 SAML 服务器自己知道的对称密钥 K 来加密。在这个协议中，由于 $Token_c$ 仅能由其发放者 A 来解读和发放，其内容可以认为是安全的。K_{rand} 为 C 生成的一次性随机密钥，由协议可知，这个密钥只是用来对密钥 K_{ca} 进行加密。K_{ca} 是 A 生成的 C 和 A 之间的密钥，通过用 C 产

生的K_{rand}加密后分配给C。K_{cs}是A生成的S和A之间的密钥，实际上在认证协议的后半部分，这个密钥也参与对C的身份认证。N_a、N_c、N_c'、N_s、N_s'为各个主体生成的Nonce，即一次性随机数。K_a为主体A的公钥，K_a^{-1}为其对应的私钥。主体C和S的密钥也使用类似的符号表示。协议中所使用的符号定义由表1进行描述。

表1 协议中的符号描述

出现的符号	符号的含义
A、C、S	主体的名称
Request	认证请求
$Cert_a$、$Cert_c$、$Cert_s$	主体的公钥证书
$Token_c$	用户C的属性断言
K_{rand}、K_{ca}、K_{cs}	主体生成的共享密钥
N_a、N_c、N_c'、N_s、N_s'	主体生成的一次性随机数
K_a、K_c、K_s	主体的公钥
K_a^{-1}、K_c^{-1}、K_s^{-1}	主体的私钥
T_a	主体A产生的时间戳

通过对基于SAML的云服务安全认证模型的研究，将该协议用消息形式表述为：

Msg1 $C\rightarrow A$ $Request_{ca}$

C向A发出认证请求，请求A的公钥证书；

Msg2 $A\rightarrow C$ $Cert_a,\{N_a,Request_{ca}\}K_a^{-1}$

A向C提供其公钥证书并用私钥签名信息，C将校验A的公钥证书上的XKMS签名；

Msg3 $C\rightarrow A$ $A,\{Cert_c,\{N_a,N_c,K_{rand},A,C\}K_c^{-1}\}K_a$

C向A请求属性断言，消息先使用C的私钥数字签名，再使用A的公钥加密，N_a、N_c是为防止重放攻击而传递的挑战信息，K_{rand}是C产生的随机的一次性密钥；

Msg4 $A\rightarrow C$ $Token_c,\{N_c+1,A,C,K_{ca}\}K_{rand}$ $Token_c=\{A,C,K_{ca},T_a,Lifetime\}K$

A验证C身份后，生成A与C之间的会话密钥K_{ca}，并向C提供属性断言，K_{ca}包含在$Token_c$令牌中并且仅由C保存，时间戳T_a用于防止重放攻击，使用仅有A知道的对称密钥K加密，防止C重建令牌；

Msg5 $C\rightarrow S$ $Token_c,Request_{cs},N_c'$

C向S发出服务请求；

Msg6 $S\rightarrow A$ $Cert_s,\{Token_c,N_c',S,C,N_s\}K_s^{-1}$

S将C的属性断言送回A验证；

Msg7 $A\rightarrow S$ $S,\{Cert_a,\{\{N_c'-1,C,S,K_{cs}\}K_{ca},C,S,N_s+1,K_{cs}\}K_a^{-1}\}K_s$

A生成C、S之间的会话密钥K_{cs}，其包含在消息中经A数字签名后发送给S，消息中的$\{N_c'-1,C,S,K_{cs}\}K_{ca}$只能由$C$解读，因此$S$将其转发给$C$；

Msg8 $S\rightarrow C$ $\{N_c'-1,C,S,K_{cs}\}K_{ca},\{N_c'+1,S,C,N_s'\}K_{cs}$

将$\{N_c'+1,S,C,N_s'\}$采用K_{cs}加密作为认证码，与S转发的A给C的消息相互验证来证明S的身份；

Msg9 $C\rightarrow S$ $\{N_s'+1\}K_{cs}$

C对S的认证响应，通过向S发送对$N_s'+1$的加密信息来证明C的身份。

3 模型检测及处理流程

3.1 模型检测与通信顺序进程

模型检测(Model Checking)将所需验证的系统及属性作为模型检测工具的输入，检测系统所有可能的状态；验证这些状态是否满足用户需要的性质。通信顺序进程CSP(Communicating Sequential Processes)可用于安全协议的描述与分析[12]。目前应用较为普遍的模型检测工具有符号模型检测SMV(Symbolic Model Checking)工具、Casper[14](a Compiler for the Analysis of Security Protocols)+FDR[13](Failures-Divergence Refinement)工具、SPIN (Simple Promela INterpreter)工具等。

3.2 模型检测工具及处理流程

由于生成FDR代码是复杂的系统工程，Lowe研究了安全协议分析编译器Casper，用户使用一定的抽象符号对安全协议及性质进行描述，制定一份描述协议的脚本(spl文件)提交给Casper，Casper就可以将此脚本编译成FDR所需的CSP代码。在本文中使用Casper和FDR对安全协议进行分析的流程为：

(1)将安全协议描述为以spl为后缀名的抽象描述文件作为Casper的输入；

(2)将Casper输出的以csp为后缀名的文件作为FDR的输入；

(3)通过FDR检测系统是否存在攻击，验证协议的安全性。

图3是Casper-FDR软件工具对安全协议的处理流程。

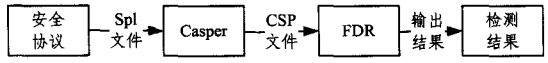

图3 Casper-FDR软件工具对安全协议的处理流程

4 云服务认证协议的CSP模型检测

4.1 云服务安全认证协议建模与初步检测

首先对本文提出的云服务安全认证协议进行建模，写出符合Casper规范的抽象spl格式的输入文件。对安全协议进行整体建模，如下：

```
#Free variables
c,s,a:Agent
nc,na,ns,nc',ns':Nonce
kca,kcs,krand:SessionKey
PK:Agent->PublicKey
SK:Agent->SecretKey
InverseKeys=(krand,krand),(kca,kca),(kcs,kcs),(PK,SK)
```

在此声明协议中出现的变量。

```
#Processes
INITIATOR(c,a,s,nc,nc',krand) knows PK(c),SK(c),PK(a),PK(s)
RESPONDER(s,a,ns,ns',kcs) knows PK(s),SK(s),PK(a)
SERVER(a,s,na,kcs,kca) knows PK(a),SK(a),PK(c),PK(s)
```

分配3个主体的角色及其初始知识。

```
#Protocol description
0. ->c:a
1. c->a:c
2. a->c:PK(a),{na,c}{SK(a)}
3. c->a:a,PK(c),{{na,nc,krand,a,c}{SK(c)}}{PK(a)}
4. a->c:{nc,a,c,kca}{krand}
5. c->s:c,nc'
6. s->a:PK(s),{nc',s,c,ns}{SK(s)}
7. a->s:s,{PK(a),{c,s,na,kcs,{nc',c,s,kcs}{kca}}{SK(a)}}
   {PK(s)}
```

8. s->c:{nc',c,s,kcs}{kca},{nc',s,c,ns'}{kcs}
9. c->s:{ns'}{kcs}

将协议的消息流程写成 spl 规定的格式。

将 spl 抽象描述文件作为 Casper 的输入，生成 csp 文件，并将该文件输入到模型检测工具软件 FDR 中。FDR 软件一直处于检测状态而无法显示结果，根据对软件运行日志的综合分析，造成该问题的原因是采用模型检测法分析云服务安全认证协议时，多个主体、多条消息之间的大量交互过程中，模型检测软件需要穷举协议的各个可能状态，由于大量的云服务与多个主体的大量消息交互，所产生的状态空间爆炸使得模型检测软件无法直接得到检测结果。

4.2 协议分割后的模型检测

针对云服务安全认证协议整体建模出现的问题，在认证协议的安全目标不变的前提下，依据安全认证协议的逻辑处理过程，按照参与主体与交互内容的差异，云服务安全认证协议按照认证及访问控制逻辑处理过程可分为两个相对独立的过程。第一个过程实现 SAML 服务器对云用户的认证与访问控制，在第一个过程完成的基础上，实现多个云服务对云用户的认证与访问控制。因此，本文把认证协议分割成为两个规模较小的组成部分，每部分包含较少的主体、消息和交互过程，从而减少云服务认证协议模型检测过程中产生的大量中间状态及交互过程。本文首先把前 4 条消息作为协议的第一部分进行安全建模及模型检测。

协议的前半部分消息如下：

Msg1 $C \rightarrow A$ $Request_{ca}$

Msg2 $A \rightarrow C$ $Cert_a, \{N_a, Request_{ca}\}K_a^{-1}$

Msg3 $C \rightarrow A$ $A, \{Cert_c, \{N_a, N_c, K_{rand}, A, C\}K_c^{-1}\}K_a$

Msg4 $A \rightarrow C$ $Token_c, \{N_c+1, A, C, K_{ca}\}K_{rand}$

其中 $Token_c = \{A, C, K_{ca}, T_a, Lifetime\}K$。

可以看出该部分的主要内容是传递密钥 K_{ca}，其检测目标也将围绕 K_{ca} 给出。写出其基本系统对应的 spl 文件：

```
#Free variables
c,a:Agent
nc,na,nc':Nonce
kca,krand:SessionKey
PK:Agent->PublicKey
SK:Agent->SecretKey
InverseKeys=(krand,krand),(kca,kca),(PK,SK)
```

声明出现的主体、Nonce 和密钥，在最后写明哪些密钥是自反的。

```
#Processes
INITIATOR(c,a,nc,nc',krand)knows PK(c),SK(c),PK(a)
RESPONDER(a,na,kca)knows PK(a),SK(a),PK(c)
```

由于只有两个主体参与，因此只有初始者和响应者。

```
#Protocol description
0. ->c:a
1. c->a:c
2. a->c:PK(a),{na,c}{SK(a)}
3. c->a:a,PK(c),{{na,nc,krand,a,c}{SK(c)}}{PK(a)}
4. a->c:{nc,a,c,kca}{krand}
```

逐条描述协议的消息。

```
#Specification
Secret(c,kca,[a])
```

在这里，将认证目标写为关于密钥 kca 的秘密性，即对 c 来说 kca 是只有 a 知道的秘密。

```
#Actual variables
Cloud,Saml,Ivo:Agent
Kca,Krand:SessionKey
Nc,Na,Nc':Nonce
InverseKeys=(Krand,Krand),(Kca,Kca)
#System
INITIATOR(Cloud,Saml,Nc,Nc',Krand)
RESPONDER(Saml,Na,Kca)
#Functions
symbolic PK,SK
#Intruder Information
Intruder=Ivo
IntruderKnowledge = {Cloud, Saml, Ivo, PK(Ivo), SK(Ivo),
PK(Cloud),PK(Saml)}
```

加入一个额外的初始者 Cloud 作为第二个系统：

```
#System
INITIATOR(Cloud, Saml, Nc, Nc', Krand); INITIATOR(Cloud,
Saml,Nc,Nc',Krand)
RESPONDER(Saml,Na,Kca)
```

加入一个额外的响应者 SAML 作为第三个系统：

```
#System
INITIATOR(Cloud,Saml,Nc,Nc',Krand)
RESPONDER(Saml,Na,Kca);
RESPONDER(Saml,Na,Kca)
```

分别检测这 3 个系统，均未发现可能的攻击，检测结果如图 4 所示。故在这一部分中密钥 K_{ca} 的秘密性得到证明。

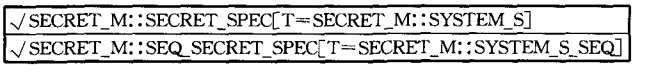

图 4 协议第一部分的检测结果

在对第一部分进行安全建模和模型检测的基础上，对协议的第二部分进行分析建模：

Msg5 $C \rightarrow S$ $Token_c, Request_{cs}, N_c'$

Msg6 $S \rightarrow A$ $Cert_s, \{Token_c, N_c', S, C, N_s\}K_s^{-1}$

Msg7 $A \rightarrow S$ $S, \{Cert_a, \{N_c'-1, C, S, K_{cs}\}K_{ca}, C, S, N_s+1, K_{cs}\}K_a^{-1}\}K_s$

Msg8 $S \rightarrow C$ $\{N_c'-1, C, S, K_{cs}\}K_{ca}, \{N_c'+1, S, C, N_s'\}K_{cs}$

Msg9 $C \rightarrow S$ $\{N_s'+1\}K_{cs}$

对其进行安全建模，如下：

```
#Free variables
c,s,a:Agent
na,ns,nc',ns':Nonce
kcs:SessionKey
PK:Agent->PublicKey
SK:Agent->SecretKey
kca:Agent->ServerKeys
InverseKeys=(kca,kca),(kcs,kcs),(PK,SK)
```

在这里认为 kca 是已经存在的密钥，因此没有像上一部分把 kca 定义成 SessionKey，而将其定义成一个 ServerKey。这里有 3 个主体参与协议交互过程。

```
#Processes
INITIATOR(c,a,s,nc')knows PK(c),SK(c),PK(a),PK(s),kca(c)
RESPONDER(s,a,ns,ns',kcs)knows PK(s),SK(s),PK(a)
SERVER(a,s,na,kcs)knows PK(a),SK(a),PK(c),PK(s),kca
#Protocol description
```

0. −>c:s
1. c−>s:c,nc′
2. s−>a:PK(s),{nc′,s,c,ns}{SK(s)}
3. a−>s:s,{PK(a),{c,s,na,kcs,{nc′,c,s,kcs}{kca(c)}}{SK(a)}}{PK(s)}
4. s−>c:{nc′,c,s,kcs}{kca(c)},{nc′,s,c,ns′}{kcs}
5. c−>s:{ns′}{kcs}
#Specification
Secret(c,kcs,[a,s])
Secret(s,kcs,[a,c])
Agreement(c,s,[kcs])
Agreement(s,c,[kcs])

安全认证目标体现出密钥 kcs 的秘密性和认证性，这两个性质对 c 和 s 来说是对等的。

#Actual variables
Cloud,Sam,Saml,Ivo:Agent
Kcs:SessionKey
Na,Ns,Nc′,Ns′:Nonce
InverseKeys=(Kcs,Kcs)
#System
INITIATOR(Cloud,Saml,Sam,Nc′)
RESPONDER(Sam,Saml,Ns,Ns′,Kcs)
SERVER(Saml,Sam,Na,Kcs)
#Functions
symbolic PK,SK,kca
#Intruder Information
Intruder=Ivo
IntruderKnowledge={Cloud, Sam, Saml, Ivo, PK(Ivo), SK(Ivo), PK(Cloud),PK(Saml),PK(Sam)}

使用 FDR 软件检测发现第二阶段的安全目标得到验证，如图 5 所示。

√SECRET_M::SECRET_SPEC[T=SECRET_M::SYSTEM_S]
√SECRET_M::SEQ_SECRET_SPEC[T=SECRET_M::SYSTEM_S_SEQ]
√AUTH1_M::AuthenticateRESPONDERToINITIATORAgreement_na_nb
√AUTH2_M::AuthenticateINITIATORToRESPONDERAgreement_kab

图 5　协议第二部分的检测结果

由本文的模型检测实验结果可知，当云服务安全主体的个数、消息条数和交互次数较多时，由于中间状态及交互过程过多难以得出模型检测结果，甚至会产生状态空间爆炸问题，这是对云服务安全认证协议进行整体建模及检测时遇到的瓶颈问题。针对该问题，本文提出在安全认证目标不变的前提下可对云服务安全认证协议按照认证主体及逻辑处理过程进行两阶段分步检测处理，并且得到了预期的安全检测结果，安全认证目标得以验证。

结束语　本文针对基于 Web Services 的云服务提出了基于 SAML 协议的云服务器跨域安全认证方案，云用户只需要进行一次身份认证即可接入多种云服务，适用于跨域云服务用户认证。使用安全协议分析软件 Casper 对本文所提出的云服务安全认证协议进行分析，并结合模型检测工具软件 FDR 对安全协议进行模型检测。实验结果表明，在安全认证目标不变的前提下，对云服务安全认证协议依据参与主体及交互过程进行分阶段拆分后，使用模型检测工具软件可有效避免安全协议检测遇到的状态空间爆炸问题，达到了协议的安全目标。本文贡献主要体现在两方面：

1)提出了基于 SAML 协议的云服务跨域安全认证方案，该方案有效集成了 Web Services、SAML、XKMS 和 PKI 等 Web Service 信息安全技术，并对云服务安全认证协议进行了抽象的模型描述。

2)采用 Casper 和 FDR 模型检测软件对本文提出的云服务跨域安全认证协议进行了分阶段的形式化分析与安全模型检测，解决了云服务认证协议整体建模与验证时的状态空间爆炸问题。

实验表明，本文提出的基于 SAML 协议的跨域云服务安全认证方案安全可靠，实现了云服务安全认证目标。

参 考 文 献

[1] Mishra A, Jain R, Durresi A. Cloud computing: networking and communication challenges[J]. IEEE Communications Magazine, 2012,50(9):24-25

[2] Grosse E, Howie J, Ransome J, et al. Cloud Computing Roundtable[J]. IEEE Security & Privacy, 2010,8(6):17-23

[3] Feng Deng-guo, Zhang Min, Zhang Yan, et al. Study on Cloud Computing Security [J]. Journal of Software, 2011,22(1):71-83 (in Chinese)
冯登国，张敏，张妍，等. 云计算安全研究[J]. 软件学报，2011,22(1):71-83

[4] Armbrust M, Fox A, Griffith R, et al. Above the Clouds : A Berkeley View of Cloud Computing[EB/OL]. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[5] Nordbotten N A. XML and Web Services Security Standards [J]. IEEE Communications Surveys & Tutorials, 2009,11(3):4-21

[6] Harding P, Johansson L, Klingenstein N. Dynamic Security Assertion Markup Language: Simplifying Single Sign-On[J]. IEEE Security & Privacy, 2008,6(2):83-85

[7] Marques Jr A P, Ravn A P, Srba J, et al. Model-checking web services business activity protocols[J]. International Journal on Software Tools for Technology Transfer, 2013,15(2):125-147

[8] Sheng Q Z, Maamar Z, Yahyaoui H, et al. Separating operational and control behaviors: A new approach to Web services modeling[J]. IEEE Internet Computing, 2010,14(3):68-76

[9] Zech P. Risk-based security testing in cloud computing environments[C]// 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2011:411-414

[10] Lin Chuang, Su Wen-bo, Meng Kun, et al. Cloud computing security: architecture, mechanism and model evaluation[J]. Chinese Journal of Computers, 2013,36(9):1765-1784(in Chinese)
林闯，苏文博，孟坤，等. 云计算安全：架构，机制与模型评价[J]. 计算机学报，2013,36(9):1765-1784

[11] Chen Hong-song, Bhargava B, Fu Zhong-chuan. Multilabels-Based Scalable Access Control for Big Data Applications[J]. IEEE Cloud Computing, 2014,1(3):65-71

[12] Dalal A, Jeff K, Alessandra R, et al. Elaborating Requirements Using Model Checking and Inductive Learning[J]. IEEE Transactions on Software Engineering, 2012,39(3):361-383

[13] Gavin L. Breaking and fixing the Needham Schroeder public-key protocol using FDR[J]. Software Concepts and Tools, 1996,17:993-999

[14] Gavin L. Casper: a compiler for the analysis of security protocols [J]. Journal of Computer Security, 1998,6(1):53-58