基于RDF4S的一种轻量语义Web服务组合

黄蓓蓓[1] Philip C-Y Sheu[1,2] 应 时[1] 张 韬[1] 崔 华[1] 张桂刚[1]
(武汉大学软件工程国家重点实验室 武汉 430072)[1]
(加州大学尔湾分校电子工程与计算机科学系 美国加州 92697)[2]

摘 要 首先简要介绍语义Web服务资源描述框架(RDF4S),在此基础上提出语义Web服务包装机制,语义Web服务经包装后,分别通过组合接口和调用接口为组合语义Web服务的设计与调用提供支持。然后分别详细介绍组合语义Web服务的设计存储和调用执行:通过图形化的设计工具按照语义Web服务的3种基本逻辑关系设计组合结构图,然后将该图映射到组合结构信息表存储,组合语义Web服务执行时根据结构信息表生成语义Web服务代理对象和相应的语义消息,再根据语义消息传递机制调用Web服务实体。最后用一个实例展示组合语义Web服务执行时的消息传递,试验表明该方法有效。

关键词 服务资源描述框架,语义Web服务包装,组合结构信息表,语义消息,语义消息传递机制
中图法分类号 TP301 **文献标识码** A

Lightweight Semantic Web Service Composition Based on RDF4S

HUANG Bei-bei[1] Philip C-Y Sheu[1,2] YING Shi[1] ZHANG Tao[1] CUI Hua[1] ZHANG Gui-gang[1]
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)[1]
(Department of EECS, University of California, Irvine, CA 92697, USA)[2]

Abstract With the increment of the resource of Web Services, many effective approaches have been applied in Web Service composition. As few Web Services is with adequate information for Web Service composition and commonly the composition approaches adapt to professional programmers only, a lightweight composition approach based on RDF4S was proposed. The paper first introduced Resources Description Frame-work for Web Service (RDF4S) and the mechanism for packaging Web Service. RDF4S was used to describe the Web Services, and packaging the Web Services according to RDF4S will supply the Web Service composers and the composition engine with the information for composition. Then a Semantic Web Service composition system based on RDF4S was brought forward. The system consists of a composition designer and a execute engine. The composition designer is a UI tool used by Web Service composers to design the composition structure according to several relation logic rules, and the structure will be stored in a Composition Structure Information Table; the execute engine restores the composition structure from the Composition Structure Information Table, and executes the semantic composed Web services according to the semantic message transport mechanism. Finally, an instance was created for testing the Semantic Web Service composition system, and the results show the approach is effective and flexible.

Keywords Resource description framework for Web service, Semantic Web service packaging, Composition structure information table, Semantic message, Semantic massage transport mechanism

1 引言

语义Web服务技术的出现使连接、管理和组合各种企业内和企业间的服务资源能以一种统一方式互联互操作以支持业务过程自动化成为可能。通过组织协调多个语义Web服务来构造新的Web服务资源,可以提高服务组件的可重用性和利用率,减少开发时间,降低开发成本。

随着语义Web服务资源不断增多,许多有效的组合Web服务资源的方法被不断提出,国外典型的有基于动态环境编排的SELF-SERV[1],项目METEOR-S中基于MWSCF框架的动态组合方法[2];国内的有基于领域本体的服务动态组合方法[3],还有基于SOA架构的Web服务组合方法[4]。

然而,现有Web服务或语义Web服务本身不具备可组合能力,为了实现语义Web服务的组合,往往通过编程的方式来为其提供可组合能力。例如,使用BPEL4WS或BPM来编排服务以实现服务组合[7,8]。又如,使用OWL-S为组合服

到稿日期:2008-07-09 返修日期:2009-02-03 本文受国家高技术研究发展计划(863)(2006AA01Z168)资助。
黄蓓蓓(1980-),男,博士研究生,研究方向为语义计算、语义软件工程、自然语言理解,E-mail: hbb21st@163.com;Phillip C-Y Sheu(1956-),男,博士生导师,教授,研究方向为语义计算、可信计算、语义Web数据集成、语义软件工程;应 时(1965-),男,教授,博士生导师,研究方向为语义软件工程、语义Web服务及其反射。

务定义过程本体实现对组合结构的支持[9,10]。这些组合方法并未解决服务本身不具备显式可组合能力的问题。同时，这种方法偏向底层应用，对于组合设计者要求过高。

本文为组合语义 Web 服务设计者提供一套组合服务支撑工具来支持服务的组合设计与执行。该支撑工具由组合服务设计器（以下简称设计器）和组合服务执行引擎构成。设计器根据 Web 服务包装机制对 Web 服务进行包装，还为用户提供图形化的组合服务结构设计器，用户根据 Web 服务包装后提供的相应接口设计组合服务。组合服务执行引擎根据语义消息传递机制执行组合服务。

2 服务描述模型 RDF4S

服务描述模型 RDF4S 的提出主要针对以下两个问题：

（1）语义描述力度不足。当前的各种标准仍然无法为各类 Web 资源提供充分的语义信息来支持各类系统软件和应用软件的运行；（2）缺乏基于语义信息将语义服务资源动态组合起来，形成大规模复杂应用软件的方法、技术和支撑平台。“它通过定义一种新的服务资源语义描述方法，将 Web 服务封装成语义信息丰富的自组织、自描述、可动态发现、可动态组合的自治的软件实体。”

RDF4S 描述的服务包括以下 4 个部分：基本信息、功能信息、QoS 信息、实现信息，如图 1 所示。它们都为组合过程提供必要的支持。

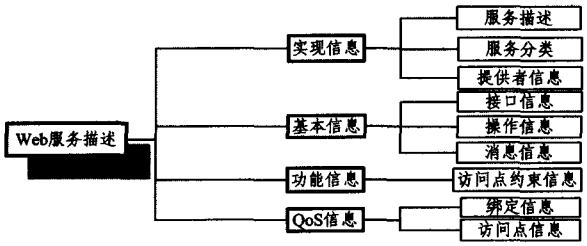

图 1 RDF4S 模型

3 语义 Web 服务组合的架构

语义 Web 服务组合工具由组合服务设计器（以下简称设计器）和执行引擎两部分组成，总体框架如图 2 所示。组合服务设计者通过设计器将所需的语义 Web 服务按所需组合起来，生成组合信息结构表存储，然后对组合后的 Web 服务进行包装并发布为新的语义 Web 服务资源。调用组合语义 Web 服务时，一方面，组合服务执行引擎从组合结构表中取出即将被调用的服务，另一方面，语义消息解析器接收调用消息并将其解析放入消息池，执行引擎将对所有消息的状态进行监听，根据消息所处的状态来调用某一具体的语义 Web 服务。

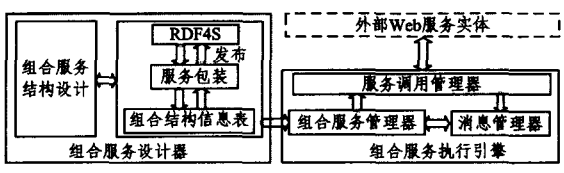

图 2 语义 Web 服务组合工具框架

4 语义 Web 服务包装

语义 Web 服务提供的描述信息可以从两个层次进行组织以实现对 Web 服务包装。第一层是服务层，它是对服务的整体描述；第二层是接口层，它是对服务功能的描述，每一个接口对应服务的一个操作。不同角色使用的接口不同，获得的信息也不同，组合设计者只能获得组合信息，而执行引擎可以获得调用信息。

4.1 组合语义信息抽取

从语义 Web 服务描述信息（RDF4S）中抽取用于组合的语义信息，利用这些信息对每个服务进行包装，包装后的每个语义服务都是一个可组合单元，结构如图 3 所示，每个单元的可视化编程接口（组合接口）都呈现给组合设计者，供其设计组合服务。此外，组合设计完成后，对组合语义 Web 服务进行包装，使其具有与其它语义 Web 服务相同的形式，实现对语义 Web 服务资源的组合建设，从而达到语义 Web 服务资源增值的目的。

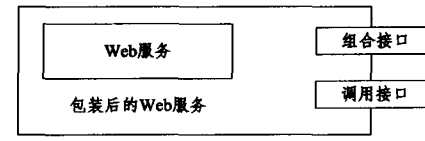

图 3 语义 Web 服务资源包装

RDF4S 描述信息给出了语义 Web 服务资源关于其位置、功能、性能、接口、输入消息、输出消息等方面的语义信息，通过标注将这些语义信息映射为本体库中规约的概念。表 1 列出了支持组合的语义信息，表 2 列出了支持服务调用的语义信息，下面将根据这两张表从 RDF4S 语义 Web 服务资源描述文件中抽取信息。

表 1 支持组合的语义信息

名称	描述	用途
服务名称 SerName	在服务空间内唯一标识该服务	服务标识
服务描述 SerDescription	描述该服务所提供功能	服务选择
输入消息类型 InMsgType	标示该服务输入消息类型	输入消息标识
输出消息类型 OutMsgType	标示该服务输出消息类型	输出消息标识
消息交换模式 MsgPattern	四种消息交换模式	消息模式标识

表 2 支持调用的语义信息

名称	描述	用途
服务名字空间 SerNamSpace	声明服务的名同空间	名字空间标识
服务绑定端口 SerLocation	声明服务描述文件中规约的绑定端口位置信息	位置标识
服务操作名称 OpeName	声明该服务所提供功能	功能选择
输入消息类型 InMsgType	标示该服务输入消息类型	输入消息标识
输出消息类型 OutMsgType	标示该服务输出消息类型	输出消息标识
消息风格 MsgStyle	消息的协议类型	消息的类型
消息传输协议 MsgTraProtocol	消息传输时使用的传输协议	传输协议

4.2 服务包装接口

包装后的语义 Web 服务为组合服务设计者提供了可组合接口，该接口对设计者可见；还提供了组合设计者不可见的语义 Web 服务可调用编程接口（调用接口）。设计者通过连结服务的输入/输出消息接口并设定相关消息信息即可设计

组合。语义 Web 服务用于组合的语义信息将被描述为如下形式：

InterfaceName（〈InMsgType，OutMsgType〉，Description，MsgPattern）

其中，输入消息类型 InMsgType 和输出消息类型 OutMsgType 的序偶将被标识为可组合接口，即〈InMsgType，OutMsgType〉。而序偶中这两者的顺序由该消息交换模式唯一确定。对于 Only-In 消息交换模式的服务，那么其可组合接口将被定义为〈InMsgType，Trigger〉，其中 Trigger 是自动生成的触发消息。同样，对于 Only-Out 消息交换模式的服务，其可组合接口将被定义为〈Trigger，InMsgType〉。而对于 Out-In 消息交换模式的服务，其可组合接口被分解为两个可组合接口的序偶，即：

〈〈Trig-ger，OutMsgType〉，〈OutMsgType，InMsgType〉〉

由此，语义 Web 服务的可组合接口将唯一确定该服务在组合过程中将被执行的调用接口。而服务的可调用接口表示为如下模式：

InterfaceName（NamSpace，Location，InMsgType，OutMsgType，MsgStyle，MsgProtocol，MsgTraPortocol，MsgPattern）

5 组合语义 Web 服务设计

5.1 组合语义 Web 服务设计

组合语义 Web 服务设计指组合服务设计者通过使用组合服务设计工具设置服务之间的组合逻辑关系。本文所涉及的服务间的逻辑关系有 3 种：顺序、并发和选择。由于语义 Web 服务都是无状态实体，可以通过消息传递关系实现对 Web 服务状态的控制，消息传递关系共 3 种，分别是消息映射、消息分解和消息合并，如图 4 所示。

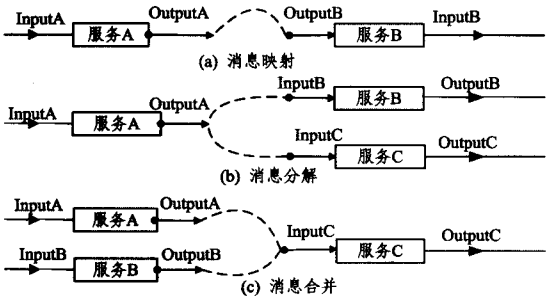

图 4 消息传递关系

组合语义 Web 服务设计者（以下简称设计者）通过操纵图形化组合设计工具设定参与组合的 Web 服务及其之间的组合关系。组合设计者通过以下 3 个步骤完成组合语义 Web 服务间逻辑关系的设定：

1. 设计者根据包装后的语义 Web 服务的可组合接口选择将参与组合的 Web 服务。

2. 设计者根据 Web 服务提供的可组合接口，利用图形化工具设定 Web 服务组合接口之间的关联关系。

3. 设计者定义具有组合关系的可组合接口之间的消息映射关系，该步骤表明设计者只能设定可组合接口之间的映射关系，调用接口间的消息映射关系是无法在组合设计阶段确定的。

5.2 组合结构信息表

组合结构信息指原子服务及其相互传递的消息之间形成的网络结构，在该网络中，每一个结点代表一个服务实例，结点之间的边表示服务之间具有消息传递关系，服务之间通过消息进行互操作。组合服务中每个语义服务相对于该组合服务而言都是一个原子服务，对应一个结点，事实上该原子服务也可能是一个组合服务，因此“组合”或“原子”只是在一个组合服务中相对而言的。当组合服务运行时，组合服务支撑环境根据组合结构提取语义 Web 服务信息调用执行。

组合结构信息表用于记录组合结构信息，其表结构如下：（senderID，recei-verID，megID），senderID 和 receiverID 分别为发出消息的语义服务编号和接收消息的语义服务编号，megID 为语义消息编号。该表中的每一个元组定义了参与组合的 Web 服务之间的交换消息。同样，表中也记录了组合服务本身与内部服务之间的交换消息，为此提供两个特殊服务，称之为虚拟服务，一个负责将组合服务接收到的消息解析转发给一个或多个原子服务，另一个负责将最后原子服务的输出消息合并成一条消息返回给组合服务调用者。

当组合服务设计者完成组合服务的结构设计后，该组合服务的内部原子服务及其传递的消息被映射到组合结构信息表中，供以后组合语义 Web 服务调用执行使用。

6 组合语义 Web 服务执行

组合服务的执行在组合语义 Web 服务执行引擎（简称执行引擎）中完成。执行引擎由 3 大部分组成：组合服务管理器、消息管理器和服务调用管理器，其体系结构如图 5 所示。组合服务管理器根据组合结构信息表创建并初始化语义 Web 服务代理对象、语义消息等；消息管理器则根据设计者制定的组合关系管理语义消息的传递；服务调用管理器负责管理服务代理对象与网络上 Web 服务实体的通信。

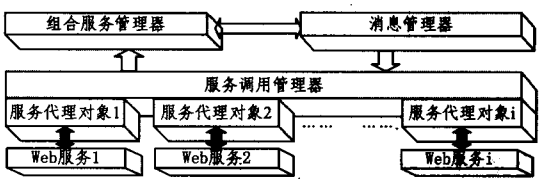

图 5 组合引擎体系结构

6.1 语义消息和语义消息项

语义消息是信息传递的载体，它形式化为二元组：（msgEnvelope，msgData，msgState）。其中，msgEnvelope 是消息信封，包含了消息传输所需协议信息；msgData 是消息数据，是消息所传输内容；msgState 是消息状态，用来指示该消息的所有消息项是否都被实例化，若都被实例化则该值为 true，默认值为 false。消息信封由消息标识符、消息发送者的标识符、消息接收者的标识符以及消息传输协议构成；消息数据包含构成该消息的语义消息项（msgItem）集合和消息的语义类型（semType）。语义项是消息传递的最基本语义单元，如起讫时间。

6.2 语义 Web 服务代理

分布在网络上的各个 Web 服务实体由相应的语义 Web 服务代理调用，每个原子服务对应一个语义 Web 服务代理实例。当组合服务执行时，首先由组合服务管理器根据组合结构信息表生成并初始化各个语义 Web 服务代理实例，然后这些代理实例根据语义消息传递机制组织消息传递的路径，组合服务接收消息后根据该路径将语义消息传递到各个语义

Web 服务代理实例。

逻辑上，原子服务之间的通信表现为语义消息传递；物理上，执行引擎中语义 Web 服务之间的通信则是间接依靠语义 Web 服务代理对象之间传递 SOAP 消息完成的，详细过程如图 6 所示。

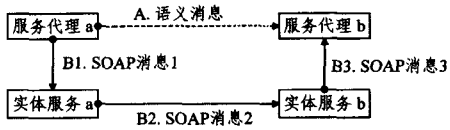

图 6　语义 Web 服务消息通信

6.3　语义消息传递机制

语义消息的传递路径由语义消息传递机制决定。消息状态改变引起语义消息的传递，当一条语义消息所包含的所有语义项均被实例化，则该语义消息状态被设置为“Ready”，消息管理器将判断语义 Web 服务的所有输入消息是否都处于“Ready”状态，当一个语义 Web 服务所有的输入消息状态均为“Ready”时，该服务即将被调用执行，当该服务执行完毕后组合服务管理器根据组合结构信息表查找其后继服务，若查找到，则向后继服务发出语义消息，否则，向组合服务调用者返回该消息。

假设从组合结构信息表中得到的组合语义 Web 服务 W 由原子服务 $w_1, w_2, \cdots, w_i$ 组成，语义消息为 $m_1, m_2, \cdots, m_k$。现将由原子服务和语义消息构成组合语义 Web 服务图 G，原子服务为图中的结点，而语义消息为连接这些结点的有向边。为了减少扫描判断服务输入消息状态的次数，可以采用类似拓扑排序的方法，将所有入度为 0 的结点（服务）加入临时集合 TempSet，执行时从该集合中取出一个或多个原子服务进行调用，服务执行完毕的输出消息发送到后继服务 e. postService，该算法用伪代码描述如下：

```
1. For each element e in Graph G
2.    If e. in_degree = 0
3.       TempSet. add (e)
4.    End If
5. End For
6. While TempSet is not empty
7.    For each element e in TempSet
8.       If e. in_message is ready
9.          call e. service
10.         delete e from TempSet
11.         delete e from the Graph
12.         If e. postService is not null
13.         and e. postService. in_degree = 0
14.            TempSet. add(e. postService)
15.         End If
16.      End If
17.   End For
18. End While
```

7　测试实例

结合目前承担的 863 项目，以奥运旅游电子商务为背景设计了 31 类语义 Web 服务，涉及国内（国际）酒店服务、国内（国际）航班服务、火车营运服务、城市链旅行计划服务等，其中城市链旅行计划和酒店航班预定等语义服务为组合语义服务，作为服务提供商提供的一类增值服务。我们为每一类服务制作 2～3 个 Web 服务实体，将其部署到网络上，并发布其 RDF4S 描述文档。

以城市链旅行计划组合语义服务为例，该服务涉及 4 个原子服务，分别是国内酒店查询服务、国内航班查询服务、国内酒店预定和国内航班预定服务，编号分别为 1，2，3 和 4。编号 I 和编号 O 服务为虚拟服务，它们由组合服务管理器提供，不调用 Web 服务实体，仅仅提供发送消息和接收消息的功能。服务 m1……m7 表示语义消息，m4 表明必须根据国内航班查询服务（编号 2）的输出结果来确定国内酒店预定服务（编号 3）的输入，由这些服务和消息构成的组合结构图如图 7 所示。

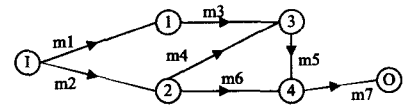

图 7　组合语义 Web 服务实例

试验环境为 CPU：P4，OS：WinXP，服务实体部署在 GlassFish Web 应用服务器上，组合服务设计器截面如图 8 所示，执行引擎 UML 图如图 9 所示，执行后输出服务序号为 1，2，3，4，由于服务 1 和 2 是并行执行，故其顺序随机生成。

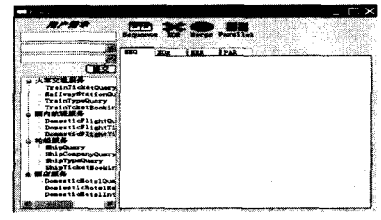

图 8　组合服务设计器截图

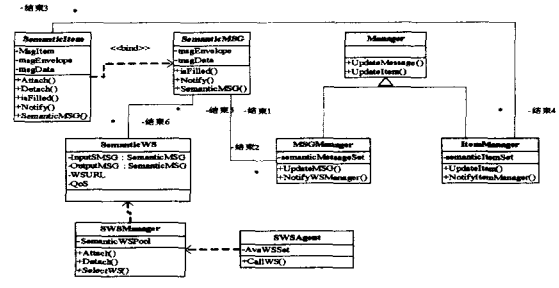

图 9　执行引擎 UML 图

结束语　本文首先简要介绍语义 Web 服务资源描述框架（RDF4S），在此基础上为组合服务的设计与调用提供语义 Web 服务包装，介绍了调用接口和组合接口，然后详细介绍组合语义 Web 服务的设计以及组合服务的调用执行机制。最后用一个实例展示组合语义 Web 服务的设计与调用执行。

参 考 文 献

[1] Benatallah B, Sheng Q Z, Dumas M. The Self-Serv Environment for Web Services Compos-ition[J]. IEEE Internet Computing, 2003, 7(1): 40-48

[2] Patil A, Oundhakar S, Sheth A, et al. METEOR-S Web service Annotation Framework[A]//Proc. of the 13th Intl World Wide Web Conf[C]. New York: ACM Press, 2004: 553-562

[3] Wang M D Z, Du X Y, Wang S. Dymamic composit - on of Web services based on domain ontology[J]. Chinese Journal of Computers, 2005, 28(4): 644-650

[4] Gao Yan, Zhang Shao - xin, Zhang Bin, et al. SOA Based Web Service Composition System [J]. Journal of Chinese Computer Systems, 2007, 28(4): 729-733

[5] Xie Dan, Ying Shi, et al. A Resource DescriptionFramework for Service[C]// Wireless Communications, Networking and Mobile

Computing, 2007. WiCom 2007, Wuhan, 2007
[6] Liang Wen-yau, Huang Chun-che, Chuang Horng-fu. The design with object (DwO) approach to Web services composition[J]. Computer Standards & Interfaces, 2007(29): 54-68
[7] Meng S, Arbab F. Web Services Choreograph and Orchestration in Reo and Constraint Automata[C]// AC'07. Seoul, Korea, March 2007
[8] Qiu Zongyan, Zhao Xiangpeng, Cai Chao, et al. Towards the Theoretical Foundation of Choreography[C]// IW3C2, 2007. Banff, Alberta, Canada, 2007
[9] Agarwal S, Handschuh S, Staab S. Annotation, composition and invocation of semanticweb services[J]. Web Semantics: Science, Services and Agents on the World Wide Web, 2004(2): 31-48
[10] Charif Y, Sabouret N. An Overview of Semantic Web Services Composition Approaches[J]. Electronic Notes in Theoretical Computer Science, 2006, 146: 33-41

(上接第 107 页)

标签名为 0009(如图 4 倒数第二行所示),说明识别正确。将该交互自动进行 1000 次,每次都能够识别正确。

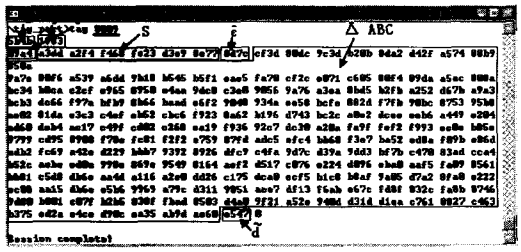

图 3 标签输入输出数据

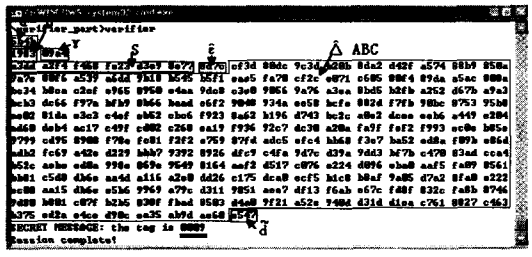

图 4 读写器输入输出数据

5 抗各种攻击的能力

本协议能够抗各种 RFID 中的常见攻击。

1. 跟踪:各次交互的 α',β',γ 和 F_j' 是动态变化的,根据式(3),攻击者只有获得 $\delta_{j,\kappa}^{(1)}$ 才知道密码值之间的关联性,$\delta_{j,\kappa}^{(1)}$ 不易获得。

2. 非法标签攻击,包括:

1)重放攻击:攻击者录制了标签发送给读写器的数据,并在以后伪装合法标签重放该数据。对本协议该攻击是无效的,因为一次成功交互以后的密码会更新,旧的密码值不再有效。

2)克隆标签:克隆标签发送的不正确的 α' 会被读写器拒绝。即使 α' 正确,也难以正确给出 γ。

3. 非法读写器攻击:非法读写器不能够就 α' 给出正确的 β',会被标签拒绝。

4. 非法读写器主动扫描标签信息:该攻击方法只能够获得当前 α',并不能够单独构成安全威胁。

结束语 在 RFID 安全的研究上,本文选择从第一类安全协议入手,该类协议只包含异或和简单的逻辑控制,非常适合在 RFID 上实现。本文在指出该类协议中的最小限度密码算法的弱点后,提出了 3 点改进方法。改进后的协议需通过强力攻击破解,克服了原协议能够根据交互数据直接计算密码值的缺点。实验结果表明,改进后的协议的破解需要更多的时间、更多的记录条数,同时改进后的协议对硬件需求的增加并不多。该研究成果将有助于在低成本的 RFID 芯片上实现较高安全性的 RFID 安全协议。

参 考 文 献

[1] Juels A. RFID Security and Privacy: A Research Survey[J]. IEEE Journal on Selected Areas in Communications, 2006, 24(2): 381-394
[2] 中华人民共和国科学技术部等十五部委. 中国射频识别(RFID)技术政策白皮书. 2006-06-09[OL]. http://www.eetchina.com/ARTICLES/2006JUN/PDF/CHINARFIDWHITEPAPER.PDF
[3] Feldhofer M, Dominikus S, Wolkerstorfer J. Strong Authentication for RFID Systems Using the AES Algorithm[C]// Cryptographic Hardware and Embedded Systems— CHES 2004—6th Int'l Workshop, LNCS 3156. Springer, 2004: 357-370
[4] Sarma S E, Weis S A, Engels D W. Radio-frequency-identification security risks and challenges[J]. CryptoBytes, 2003, 6(1)[OL]. http://www.rsasecurity.com/rsalabs/cryptobytes/CryptoBytes_March_2003_lowres.pdfJHJsearch=%22Radio-frequency-identification%20security%20risks%20and%20challenges%22
[5] Juels A, Pappu R. Squealing Euros: Privacy protection in RFIDenabled banknotes[C]// R. Wright, ed. Financial Cryptography '03, volume 2742 of Lecture Notes in Computer Science. Springer-Verlag, 2003: 103-121
[6] Sarma W S, Rivest R, Engels D. Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems[C]// 1st Intern. Conference on Security in Pervasive Computing (SPC). 2003: 454-469
[7] Ohkubo M, Suzuki K, Kinoshita S. Cryptographic Approach to Privacy-friendly Tags[C]// RFID Privacy Workshop. MIT, 2003
[8] Avoine G, Oechslin P. A Scalable and Provably Secure Hash Based RFID Protocol[C]// the 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec), 2005[OL]. http://lasecwww.epfl.ch/pub/lasec/doc/AO05.pdfJHJsearch=%22filetype%3Apdf%20A%20Scalable%20and%20Provably%20Secure%20Hash%20Based%20RFID%20Protocol%22
[9] Dimitriou T. A Lightweight RFID protocol to protect against Traceability and Cloning attacks[C]// IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks, SECURECOMM 2005[OL]. http://www.ait.edu.gr/faculty/T_Dimitriou_files/RFID-securecomm05.pdfJHJsearch=%22filetype%3Apdf%20A%20Lightweight%20RFID%20protocol%20to%20protect%20against%20Traceability%20and%20Cloning%20attacks%22
[10] 周永彬,冯登国. RFID 安全协议的设计与分析[J]. 计算机学报, 2006, 29(04): 581-589
[11] Juels A. Minimalist Cryptography for Low-Cost RFID Tags[C]// Security in Communication Networks-Proc. 4th Int'l Conf., LNCS 3352. Springer, 2004: 149-164[OL]. http://www.rsasecurity.com/rsalabs/node.asp?id=2033
[12] Menezes A J, van Oorschot P C, Vanstone S A. Handbook of Applied Cryptography[M]. CRC Press, 1996: 21