分类号：TP311　　密　　级：公 开
U D C ：　　单位代码：10424

学 位 论 文

基于 HTTP 的物联网终端与服务器安全通信研究与实现

王吉豪

申请学位级别：硕士学位　　专业名称：电路与系统
指导教师姓名：崔 建 明　　职　　称：副 教 授

山 东 科 技 大 学
二零一六年四月

论文题目：

基于 HTTP 的物联网终端与服务器安全通信研究与实现

作者姓名：王 吉 豪　　入学时间：2013 年 9 月

专业名称：电路与系统　　研究方向：嵌入式系统及应用

指导教师：崔 建 明　　职　　称：副　教　授

论文提交日期：2016 年 4 月

论文答辩日期：2016 年 6 月

授予学位日期：

RESEARCH AND IMPLEMENTATION OF SECURE COMMUNICATION BETWEEN IoT CLIENT AND SERVER BASED ON HTTP

A Dissertation submitted in fulfillment of the requirements of the degree of

MASTER OF PHILOSOPHY

from

Shandong University of Science and Technology

by

Jihao Wang

Supervisor: Associate Professor Jianming Cui

College of Electronics, Communication and Physics

April 2016

声　　明

本人呈交给山东科技大学的这篇硕士学位论文，除了所列参考文献和世所公认的文献外，全部是本人在导师指导下的研究成果。该论文资料尚没有呈交于其它任何学术机关作鉴定。

硕士生签名：

日　　期：

AFFIRMATION

I declare that this dissertation, submitted in fulfillment of the requirements for the award of Master of Philosophy in Shandong University of Science and Technology, is wholly my own work unless referenced of acknowledge. The document has not been submitted for qualification at any other academic institute.

Signature:

Date：

摘 要

随着物联网技术的发展，物联网终端接入网络使用的协议多种多样。HTTP 协议具有灵活、简单、快速的优点。国内许多公司选用 HTTP 协议作为物联网终端接入网络的协议，并开发了对外开放的物联网终端接入平台。

物联网终端采用 HTTP 协议与 Web 服务器通信，一方面适合物联网终端自身硬件资源少的的特点，另一方面 Web 服务技术发展成熟，开发者可以利用现成的免费软件迅速搭建 Web 服务器。

由于 HTTP 协议明文传输的特点，针对物联网终端的安全问题，国内的大多数提供物联网终端接入平台的公司使用 USERKEY 方式的安全通信模型。该模型在一定程度上验证了数据的合法性，但也存在安全隐患。攻击者很容易发起截获、伪造、篡改等网络攻击。

本文分析了 USERKEY 方式的安全通信模型的特点。然后针对其安全性能的不足，提出了加密编码方式的安全通信模型。最后将提出的模型加以实现，来测试模型的安全性能。

关键词：物联网终端；服务器；安全通信；HTTP

ABSTRACT

With the development of IoT technology, there are a variety of protocols for IoT clients to access network. HTTP protocol is flexible, simple and fast. Many domestic companies have used HTTP protocol for IoT clients to access network, and developed open IoT platform.

That HTTP protocol is empolyed by IoT clients to communicate with the Web server is suitable for IoT clients which hardware resource is limited and is convenient for developers who can quickly build a web server using ready-made free mature software

Due to the message are transported in plaintex in HTTP protocol, the majority of domestic companies that have developed open IoT platform have used USERKEY security communication model to ensure the communication security between IoT clients and server.The model verifies the validity of the data to a certain extent, but it also has the security hidden danger. It is easy for an attacker to initiate the network attacks, such as interception, forgery, tampering, etc..

This paper analyzes the characteristics of USERKEY security communication model, then proposed a Encrypt-Encoding security communication model. Finally, the proposed model is implemented for testing the security performance of the model.

Keywords: IoT client; server; secure communication; HTTP

目 录

Contents

1 绪论

本章主要介绍课题的研究背景、研究意义、研究现状、以及研究内容，最后介绍了论文的章节安排。

1.1 研究背景和意义

1.1.1 研究背景

随着云计算[1-2]的广泛使用和物联网[3]技术的发展，二者的结合越来越凸显。物联网终端接入网络使用的协议也多种多样。HTTP 协议具有灵活、简单、快速的优点。国内许多公司选用 HTTP 协议作为物联网终端接入网络的协议，并开发了对外开放的物联网终端接入平台。例如：中国移动的设备云、Yeelink 的物联云、上海感云信息技术有限公司的传感云等等。

物联网终端采用 HTTP 协议与 Web 服务器通信，是一种“物联网终端-WEB 服务器”架构。该架构一方面适合物联网终端自身硬件资源少的的特点，另一方面 Web 服务技术发展成熟，开发者可以利用现成的免费软件迅速搭建 Web 服务器。

Web 服务器一方面面向物联网终端。物联网终端采集数据，通过 HTTP 协议往 Web 服务器上报数据，Web 服务器存储物联网终端采集到的数据。另一方面，Web 服务器面向用户。用户可以通过使用 Web 浏览器远程登录到 Web 服务器，查看相应的物联网终端上报的数据或远程控制物联网终端。

由于 HTTP 协议是一种明文传输，针对物联网终端的安全问题，国内许多提供物联网终端接入平台的公司使用 USERKEY 方式的安全通信模型。即开发者在物联网 Web 服务器上注册一个帐号，便可以获取一个 USERKEY。开发者的物联网终端与服务器通信时，通信数据中都要包含 USERKEY。Web 服务器通过对 USERKEY 的验证来确定数据的合法性。而物联网终端的其他参数，如温度信息、湿度信息、气体浓度等都为明文传输。

USERKEY 方式的安全通信模型在一定程度上验证了数据的合法性，但也存在安全隐患。例如，由于明文传输的特点，攻击者通过某种手段可以截获物联网终端与服务器的通信数据，实现对通信数据的窃听。攻击者也可以通过对截获后的数据进行修改。例如伪造高数值温度数据来引发服务器的报警，或者伪造正常数据，屏蔽物联网终端的异

常数据，使服务器失去异常报警能力。攻击者通过分析数据包，就能很容易的找到USERKEY，因此可以伪造一个假的物联网终端与服务器通信，而将真的终端移走。

综上所述，可以得出结论：USERKEY 方式的安全通信模型具有 HTTP 协议优点的同时，在安全上也存在一些不足，在一定程度上不能抵御截获、伪造、篡改等网络攻击。因此，研究出一种简单实用并可以在一定程度上抵抗以上几种网络攻击的安全通信模型显得格外重要。

1.1.2 研究意义

云计算技术的发展为物联网技术的发展奠定了平台基础，一方面，云计算技术可以为物联网终端提供稳定、可靠的接入平台，另一方面，它弹性可扩展的特点为用户降低了使用成本，并且为用户后续升级平台提供了方便，它能吸引更多的用户参与使用。而物联网终端的安全是物联网技术向大规模方向发展的必要条件。只有保证物联网终端的安全才能使物联网技术的发展进入良性循环。

通过本课题的研究可以提出一种保证物联网终端安全的一种安全模型。通过对该模型的使用，可以使云计算和物联网技术的结合更加紧密。云计算可以凭借其大数据计算能力将物联网终端采集到的数据加以处理，结合数据挖掘技术，可以建立强大的分析和预测模型，为国家的宏观调控及决策提供指导和服务，对社会发展具有广泛的意义[4]。

1.2 研究现状

目前，物联网的发展仍然是起步阶段，由于物联网时代的浪潮才刚刚开始，国内外对物联网的研究还没有形成统一的国际标准。物联网的发展需要依靠互联网的发展和物的智能化发展，它把虚拟的网络和实际物体相结合需要很多技术的支持，包括传感器技术、识别技术、数据处理技术、通信网络技术、安全隐私技术等等。2012 年以来，在传感技术、云计算和移动互联网发展的推动下，全球物联网进入实质性应用阶段。欧美日韩等国家和地区在物联网技术、应用等方面取得重要进展[5-9]。

在国内也有不少公司开发了对外开放的物联网终端接入平台。例如：中国移动的设备云、Yeelink 的物联云、上海感云信息技术有限公司的传感云等等。

由于物联网安全方面没有一个统一的国际标准，国内的大多数提供物联网终端接入平台的公司使用一种 USERKEY 方式的安全通信模型，该通信模型使用 HTTP 协议进行

数据传输，由于 HTTP 协议明文传输的特点，这种通信模型在一定程度上不能抵御截获、伪造、篡改等网络攻击。

1.3 研究内容

1.“物联网终端-WEB 服务器”架构

研究“物联网终端-WEB 服务器”架构的原理，通过对该架构的研究，为进一步研究基于 HTTP 的安全通信模型奠定基础。

2. 基于 HTTP 的安全通信模型

首先研究国内许多公司使用的 USERKEY 方式的安全通信模型。通过对它的研究发现其优点和不足，并针对其不足提出一种新的安全通信模型，加密编码方式的安全通信模型。

3. 加密编码方式的安全通信模型的实现

在对加密编码方式的安全通信模型提出的基础上，结合一个实际的应用场景，对其加以实现。验证其实际可行性。

4. 加密编码方式的安全通信模型测试

通过模拟多种攻击方法对实现后的模型进行测试，来验证模型的安全性。

1.4 论文章节安排

本文结构和各章节内容安排如下：

第 1 章：本章为论文的绪论部分。主要介绍课题的研究背景、研究意义、研究现状以及主要研究内容及章节安排。

第 2 章：本章为论文的理论基础部分。简单介绍了 HTTP 协议以及“物联网终端-WEB 服务器”架构，为下一章研究基于 HTTP 的安全通信模型奠定基础。

第 3 章：本章为论文中提出的新理论模型的核心部分。主要介绍了两种基于 HTTP 的安全通信模型。首先分析了国内许多公司使用的 USERKEY 方式的安全通信模型，总结其特点，并发现其不足。在此基础上，提出了加密编码方式的安全通信模型。并对提出的新模型进行深入的介绍及分析。

第 4 章：本章为论文中提出的新理论模型的实现部分。全方位地介绍如何对第三章提出的理论模型加以实现。包括在物联网终端上的实现、在 Web 服务器上的实现以及面向用户层的物联网终端管理系统的实现。

第 5 章：本章为论文中提出的新理论模型的测试部分。对第四章中介绍的实现结果进行测试，包括实现程序的使用方法以及模拟了各种攻击来验证新理论模型的安全性。

第 6 章： 对论文做了一个概括性的总结以及新模型存在的一些不足。并对以后的实际应用做出展望。

2 理论基础

本章为论文的理论基础部分。将简单地介绍 HTTP 协议以及“物联网终端-WEB 服务器”架构，为下一章研究基于 HTTP 的安全通信模型奠定基础。

2.1 HTTP 协议

2.1.1 HTTP 协议简介

HTTP 协议，英文全称为 Hypertext Transfer Protocol，中文翻译为超文本传输协议。它工作在 OSI 七层协议中运输层 TCP 协议之上，默认使用 80 端口。它从 20 世纪 90 年代开始使用，用于 WWW 服务器与浏览器进行超文本传输。

HTTP 协议的工作方式为请求-响应方式，基于 HTTP 协议的客户端主动向服务器发出请求，服务器收到请求后向客户端返回响应信息，客户端收到响应信息后，客户端与服务器之间的网络链接就断开了。

HTTP 也是一种无状态协议，为了保存客户端与服务器之间连接历史信息，让服务器知道当前请求是上一个客户端发送的请求，有两种方法可以实现：一种方法是在客户端保存，常用的是 Cookie 技术，第二种方法是在服务器端保存，常用的是 Session 技术，Session 的实现要依赖于 Cookie 技术，这部分内容将在以后章节中介绍[10]。

随着物联网技术的发展，物联网终端设备的多样化，HTTP 协议凭借其灵活、简单、快速优点，不仅用于传输超文本，也成为物联网终端接入网络使用协议之一。物联网终端采用 HTTP 协议与服务器通信，一方面适合物联网终端自身硬件资源少的的特点，另一方面 Web 服务技术发展成熟，开发者可以利用现成的免费软件迅速搭建 Web 服务器，并可以灵活选择开发语言来开发物联网相关应用。

2.1.2 HTTP 协议格式

HTTP 有两类报文，分别是请求报文和响应报文，请求报文用于客户端向服务器发送请求，响应报文用于服务器给客户端的回应[11]。请求报文和响应报文都是由 3 部分组成，分别为开始行、首部行、实体主体[12-16]。请求报文中的开始行称为请求行，响应报文中的开始行称为状态行。它们的报文格式如图 2.1 所示。

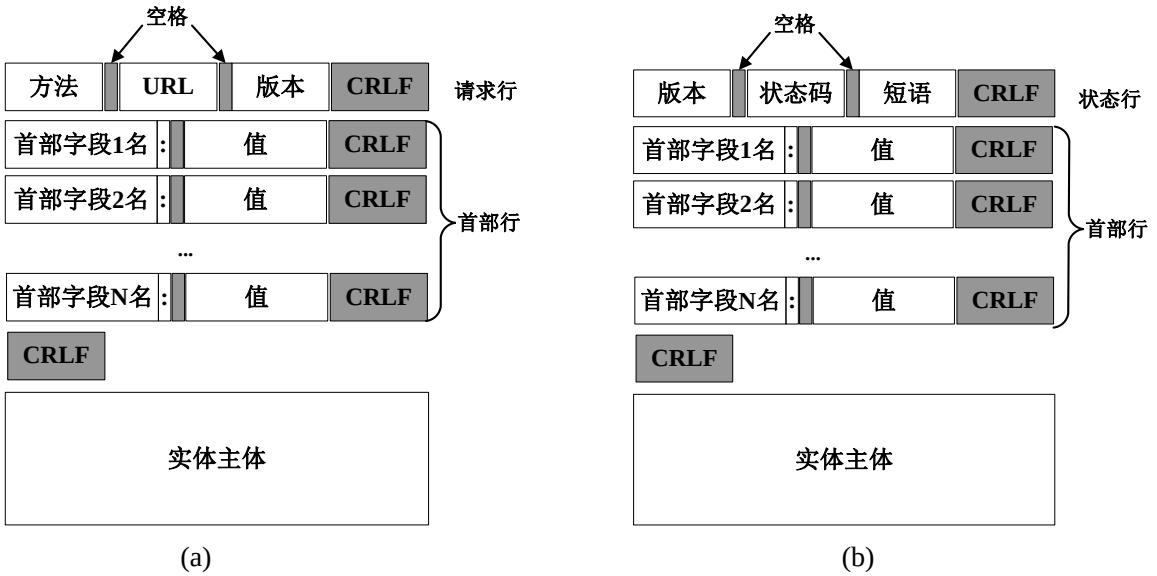

图 2.1 HTTP 报文结构：（a）请求报文； （b）响应报文

Fig. 2.1 HTTP Message structure: (a) request message ; (b) response message

1. 请求报文

请求报文的格式如图 2.1（a）所示，CR、LF 分别表示“回车”、“换行”。

请求报文的请求行包含方法、URL、版本三个字段，字段与字段之间使用空格分开。常用的方法有 GET、POST。GET 方法是最为常见的请求方法，服务器接受请求，返回客户端传入的 URL 对应的页面。POST 方法要求服务器接收来自客户端的信息，客户端的信息放到实体主体中，服务器接收客户端上传的信息长度在首部行中标识，例如，POST 实体主体中数据的长度为 10，则在首部行中加入 “Content-Length: 10”信息。URL 字段指定了要访问的路径，URL 字段中可以携带参数，例如，使用百度搜索“hello”时，请求行是这样的：“GET /s?ie=UTF-8&wd=hello HTTP/1.1”，这里面 URL 字段中的字符‘?’之后的为传入参数，可以看到传入两个参数，一个是 ie 一个是 wd，分别指定了字符的编码格式和搜索关键字。

请求报文的首部行用于说明客户端的一些信息。例如上面提到的 POST 时，指定实体主体的长度。首部行包含若干个首部字段和对应的值，它们使用“冒号”加“空格”分开，每一行首部字段后面都要有“回车”、“换行”。

请求报文的实体主体与首部行要用“回车”、“换行”分开。实体主体可以存放客户端要上传的数据。在请求报文中可以不使用实体主体。

下面是一个访问百度主页时实际的请求报文：

GET / HTTP/1.1

Host: www.baidu.com

Connection: keep-alive

Cache-Control: max-age=0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/Webp,*/*;q=0.8

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101 Safari/537.36

Accept-Encoding: gzip, deflate, sdch

Accept-Language: zh-CN,zh;q=0.8,en;q=0.6

由请求报文可以看出该请求报文仅包含请求行和首部行两部分，请求行中指定了 GET 方法和请求路径以及使用的 HTTP 协议版本。首部行中说明浏览器的信息，包括浏览器版本、可接受的编码格式、语言等等。

2. 响应报文

响应报文的格式如图 2.1（b）所示，响应报文的状态行包含版本、状态码、短语三个字段，字段与字段之间使用空格分开。常见的状态码有 200、404。200 表示请求正常，404 表示无法找到指定 URL 的资源。短语用来简单的描述请求的结果。

响应报文的首部行用于说明服务器端的一些信息。首部行包含若干个首部字段和对应的值，它们使用“冒号”加“空格”分开，每一行首部字段后面都要有“回车”、“换行”。

响应报文的实体主体与首部行要用“回车”、“换行”分开。实体主体存放客户端从服务器端请求过来的数据。在响应报文中也可以没有实体主体。

下面是一个访问不存在 URL 时的响应报文：

HTTP/1.1 404 Not Found

Server: nginx

Date: Fri, 11 Mar 2016 11:43:02 GMT

Content-Type: text/html; charset=iso-8859-1

Transfer-Encoding: chunked

Connection: keep-alive

Content-Encoding: gzip

由响应报文可以看出该响应仅包含状态行和首部行两部分，状态行中说明了传输使用的 HTTP 版本号 HTTP/1.1 和服务器的响应状态码 404 和响应短语 Not Found。

首部行中说明了服务器中的信息，包括服务器的软件信息 nginx、响应时间、传输数据的编码格式、压缩方式等等。

2.2 系统架构

“物联网终端-WEB 服务器”架构简易模型如图 2.2 所示，架构由 3 部分组成，物联网终端层、服务器层和用户应用层[17-18]。

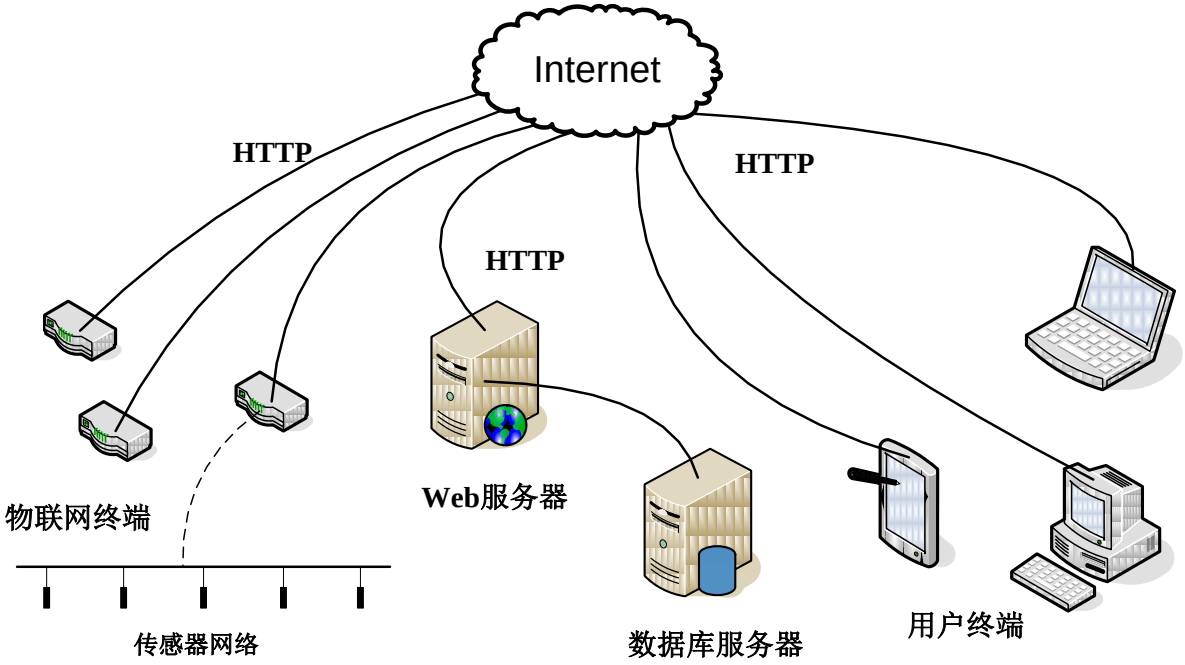

图 2.2 系统架构

Fig. 2.2 System architecture

2.2.1 物联网终端层

物联网终端[19]处在该架构的最底层，它通过有线或者无线的方式向下连接至传感器网络层[20-21]，向上连接至物联网服务器。它的任务是通过传感器采集数据，并加以处理，最后使用 HTTP 协议 GET 或者 POST 方法把处理后的数据传送到服务器。

物联网终端可以是一个带有嵌入式 linux 系统的嵌入式设备，也可以是一个带有网卡并集成网络协议栈的单片机系统，它必须具备以下能力：

1） 连接网络的能力

连接网络功能是物联网终端应该具备的最基本的功能，它包括支持 TCP/IP 协议、DNS 客户端、DHCP 客户端等功能。这些功能为物联网终端连接网络提供了条件。

2） 简单的计算能力

拥有简单的数据处理能力，使物联网终端能够有能力采集数据、组装 HTTP 报文、解析 HTTP 报文等。

2.2.2 服务器层

该架构服务器层至少由两部分组成，Web 服务器和数据库服务器，根据业务的需要，可能还要增加更多业务相关的服务器。

Web 服务器使用 HTTP 协议进行数据传输，因此 Web 服务器也被称为 HTTP 服务器。

在目前来说 Web 服务器端对客户端提供的内容服务主要包括两种类型：

(1)静态文档：静态文档是保存在 Web 服务器上预先设计好的 HTML 文档。它的内容是静态不变的，不会随着访问者的不同而不同。

(2)动态文档：动态文档的内容不是预先设计好的，而是动态的。使用动态文档的 Web 服务器通常连着一个数据库服务器，随着访问者的不同从数据库中读取的内容也不同。动态文档的生成要借助于一些脚本，常用的网站脚本开发语言有 PHP、ASP 和 JSP[22-25]。

Web 服务器为该架构的核心，它面向面向物联网终端、数据库服务器和用户。

面向物联网终端，Web 服务器接受物联网终端发出的 HTTP 请求，并把物联网终端传过来的数据加以处理。

面向用户，展现给用户的是一个物联网终端管理系统，用户可以根据账户名和密码登入管理系统，查看和管理自己的物联网终端上报上来的数据。

面向数据库服务器，Web 服务器把从物联网终端和用户那里获取的数据处理后存入数据库，或者从数据库中取出数据发给物联网终端或展现给用户等等。

数据库服务器，作为该架构的数据存储者，它用来保存物联网终端传过来的历史数据和用户的一些信息。通过物联网终端采集到的大量数据可以为以后的数据挖掘奠定基础。

2.2.3 用户应用层

用户应用层是搭建在 Web 服务器上的、面向用户使用的物联网终端管理系统，用户可以使用手机、平板或者电脑等任意一个具有 Web 浏览器的终端来访问该管理系统。通过管理系统用户可以远程查看添加过的设备的运行状态，也可以远程控制自己的设备。

2.3 本章小结

本章简单介绍了 HTTP 协议以及“物联网终端-WEB 服务器”架构，通过对 HTTP 协议以及系统架构的认识，可以为下一章研究基于 HTTP 的安全通信模型奠定理论基础。

3 基于 HTTP 的安全通信模型的研究

本章为论文中提出的新理论模型的核心部分，主要介绍两种基于 HTTP 的安全通信模型。首先将介绍和分析国内许多公司使用的 USERKEY 方式的安全通信模型，总结其特点，并发现其不足。在此基础上，提出加密编码方式的安全通信模型。并对提出的新模型进行深入的介绍及分析。

3.1 USERKEY 方式的安全通信模型

3.1.1 模型框架

USERKEY 方式的安全通信模型是国内许多公司使用的一种物联网终端与服务器通信的安全验证措施。

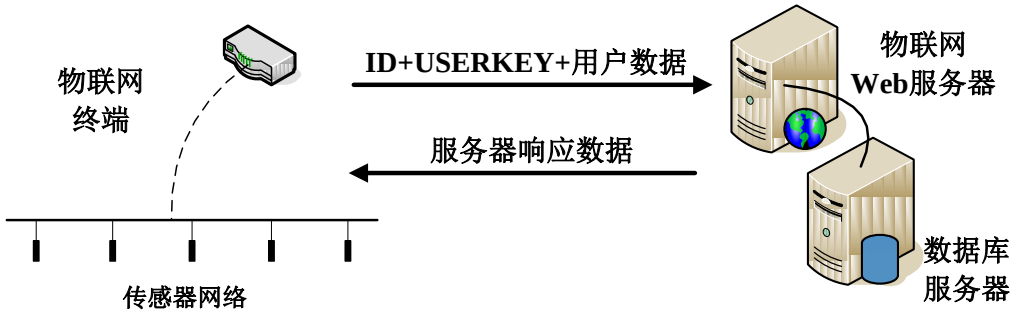

图 3.1 USERKEY 模型
Fig. 3.1 USERKEY model

图 3.1 为 USERKEY 方式的安全通信模型下的简易通信流程图。物联网终端向服务器发起 HTTP 请求时，HTTP 请求报文中包含设备 ID、USERKEY 以及与用户业务相关的数据。其中设备 ID 是物联网终端的唯一标识，USERKEY 是用户在物联网终端管理系统注册帐号后由系统分配给用户使用的，用于验证对应 ID 的设备是否是对应用户添加的设备。

在物联网终端与服务器的 HTTP 通信中，按照物联网 Web 服务器规定的格式加入设备 ID 和 USERKEY 字段就可以可使自己的设备正常接入物联网 Web 服务器，否则服务器将返回错误。

3.1.2 模型的应用实例

下面将列举国内三家公司向开发者开放的具体物联网 Web 服务器应用实例，来更好的理解该模型。

1. 中国移动的设备云

要使用中国移动提供的物联网开发平台，开发者需要为每一个接入的设备申请一个“设备 APIKey”，该设备与中国移动的设备云通信时，要将这个“设备 APIKey”放到 HTTP 协议请求报文的首部行中。在请求报文的请求行的 URL 字段中加入设备的 ID 号。例如，设备的 ID 为 73472，设备 APIKey”为“xnb6ZN01Dyp8bLGYKHEtWnpV1KEA”，下面是向物联网 Web 服务器请求读取该设备详细信息时，发起的 HTTP 的请求报文：

GET /devices/73472 HTTP/1.1

Accept-Encoding: deflate

Host: api.heclouds.com

api-key: xnb6ZN01Dyp8bLGYKHEtWnpV1KEA

下面是响应报文：

HTTP/1.1 200 OK

Date: Sat, 12 Mar 2016 08:24:21 GMT

Content-Type: application/json

Content-Length: 470

Connection: keep-alive

Server: Apache-Coyote/1.1

Pragma: no-cache

{"errno":0,"data":{"private":true,"create_time":"2015-04-15 14:58:37","keys":[{"title":"new","key":"xnb6ZN01Dyp8bLGYKHEtWnpV1KEA"}],"online":false,"location":{"lon":0,"lat":0},"auth_info":{"SYS":"VBiJJtp2gB4Db8mbdNZGiR7thOsA"},"id":"73472","datastreams":[{"unit":"摄氏度","unit_symbol":"℃","create_time":"2015-04-15 14:59:21","id":"温度","uuid":"35e1f697-7f1a-5430-a273-c446962cb1ab","tags":[""]}],"title":"传感器","desc":"传感器","tags":[]},"error":"succ"}

分析请求报文和响应报文可以发现：中国移动设备云使用的设备 ID 是一个数字，它放在了请求报文请求行的 URL 字段中，USERKEY 为“设备 APIKey”，它放在了请求报文首部行的 api-key 字段中。响应报文的实体主体是 json 格式的数据[26-28]。

2. Yeelink 的物联云

Yeelink 提供的物联网云平台，每一个注册过的用户都有一个 APIKEY，用户要使自己的物联网终端与 Yeelink 提供的物联网云平台通信时，需要把自己的 APIKEY 放到

HTTP 请求行的首部行 U-ApiKey 字段中，用户的设备 ID 则放到请求行 URL 的字段中。例如，用户的 APIKEY 为“88e3ea8b0a1cc7a89ac0e7ab43d879e8”，用户物联网终端的设备 ID 为 345516，向物联网 Web 服务器请求读取该设备详细信息时，发起的 HTTP 的请求报文如下：

GET /v1.0/device/345516/ HTTP/1.1

Host: api.yeelink.net

U-ApiKey: 88e3ea8b0a1cc7a89ac0e7ab43d879e8

Accept-Encoding: deflate

下面是响应报文：

HTTP/1.1 200 OK

Server: nginx/1.1.19

Date: Sat, 12 Mar 2016 09:09:11 GMT

Content-Type: text/html; charset=UTF-8

Connection: close

X-Powered-By: PHP/5.3.10-1ubuntu3.6

Set-Cookie: CAKEPHP=6emvui2oq36q8henvk3t0346j4; expires=Sun, 20-Mar-2016 17:09:11 GMT; path=/

P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"

Content-Length: 129

[{"title":"lunwen","about":"new","tags":"new","local":"\u9752\u5c9b","latitude":"36.020225251548","longitude":"120.13961791992"}]

分析请求报文和响应报文可以发现：Yeelink 使用的设备 ID 是一个数字，它放在了请求报文请求行的 URL 字段中，USERKEY 为“用户 APIKEY”，它放在了请求报文首部行的 U-ApiKey 字段中。响应报文的实体主体是 json 格式的数据。

3. 上海感云信息技术有限公司的传感云

同样地，要使用上海感云信息技术有限公司提供的传感云，需要用户注册一个开发者帐号，注册成功以后，用户会获得一个用户密码，用户每创建一个物联网终端的设备时会获得一个设备 ID。用户要使自己的物联网终端和传感云进行通信时，需要将用户的密码和设备 ID 放入 URL 字段中的参数中。假设用户的用户密码为 c633dab2550fec47df

b014adfa6ecc9f，用户物联网终端的设备 ID 为 56e3dc81e4b0932584df6ec4，则向物联网 Web 服务器请求读取该设备详细信息时，发起的 HTTP 的请求报文如下下：

GET /device/v1/show?ak=c633dab2550fec47dfb014adfa6ecc9f&&id=56e3dc81e4b0932584df6ec4 HTTP/1.1

Host: api.wsncloud.com

Accept-Encoding: deflate

下面是响应报文：

HTTP/1.1 200 OK

Server: nginx/1.4.4

Date: Sat, 12 Mar 2016 09:11:59 GMT

Content-Type: text/plain; charset=utf-8

Content-Length: 172

Connection: keep-alive

server-status: Ok

server-time: 1457773919912

Set-Cookie: PLAY_FLASH=;Expires=Sat, 12-Mar-2016 09:11:59 GMT;Path=/

Set-Cookie: PLAY_ERRORS=;Expires=Sat, 12-Mar-2016 09:11:59 GMT;Path=/

Set-Cookie: PLAY_SESSION=;Expires=Sat, 12-Mar-2016 09:11:59 GMT;Path=/;HTTPOnly

Cache-Control: no-cache

{"tags":"lunwen","lat":36.010445,"lng":120.13089,"city":"青岛市","id":"56e3dc81e4b0932584df6ec4","title":"lunwen","description":"lunwen","isShare":false,"sensorCount":0}

分析请求报文和响应报文可以发现：上海感云信息技术有限公司提供的传感云使用的设备 ID 是一个字符串，它放在了请求报文请求行的 URL 字段中，对应参数为 id，USERKEY 为“用户密码”，也放在了请求报文请求行的 URL 字段中，对应参数为 ak，响应报文的实体主体是 json 格式的数据。

3.1.3 模型的综合分析

通过对以上三家公司提供的物联网云平台的介绍，可以发现它们提供的云平台有一些共同的特点：

1. 每一个设备有一个独立的 ID。

每个设备有独立的 ID 可以用于区分不同的设备。物联网终端每次发起 HTTP 请求时都要包含该物联网终端的设备 ID。物联网 Web 服务器端可以根据设备 ID 迅速找到与该设备相关的数据，之后处理上报过来的数据，并做出响应。

2. 每个用户或者设备有一个接入密码。

接入密码用于验证上报数据的物联网终端是否是在物联网 Web 服务器中注册过合法的终端。这种验证机制在一定程度上保护了物联网终端和物联网 Web 服务器之间通信的安全。在本文中，我们将用户或者设备的接入密码统称为 USERKEY。

3.服务器返回 json 格式数据。

json 是一种轻量级的数据交换格式，现在已有多种语言库函数实现了对它的编解码，使用它作为数据传输一方面节省数据传输流量，另一方面方便开发者开发。

4. 传输内容透明可见。

这种特点虽然方便了开发者，开发者可以快速的调试自己的物联网终端和物联网 Web 服务器之间的通信，查看响应的结果。但这种传输内容透明可见却带来了安全隐患。

攻击者通过某种手段可以截获物联网终端与服务器的通信数据，实现对通信数据的窃听。攻击者也可以对截获后的数据进行修改。例如伪造高数值温度数据来引发服务器的报警，或者伪造正常数据，屏蔽物联网终端的异常数据，使服务器失去异常报警能力。攻击者通过分析数据包，就能很容易的找到 USERKEY，因此可以伪造一个假的物联网终端与服务器通信，而将真的终端移走。

3.1.4 结论

通过以上几小节的介绍和分析可以得出结论： USERKEY 方式的安全通信模型具有开发方便、使用简单的特点，但其安全性不高，在一定程度上不能抵御截获、伪造、篡改等网络攻击，不能适应将来物联网终端大规模部署的发展需要。因此，在 USERKEY 方式的基础上设计出一种既能保留该方式的优点又能在一定程度上抵御以上几种网络攻击的安全通信模型格外重要。

3.2 加密编码方式的安全通信模型

通过对 USERKEY 方式的安全通信模型的介绍与分析，发现该方式具有开发方便、使用简单的特点，但其带来的安全问题不能满足将来物联网终端大规模部署的发展需要，因此，本小节引入加密编码方式的安全通信模型，在保留 USERKEY 方式的安全通信模型优点的基础上，很大程度上克服其缺点，为物联网终端大规模部署奠定基础。

3.2.1 模型框架

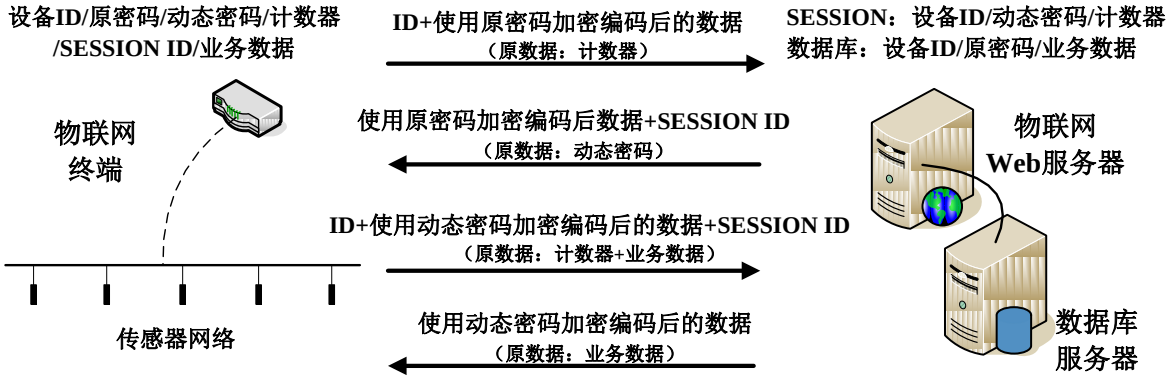

图 3.2 加密编码模型

Fig. 3.2 Encrypt-Encoding model

图 3.2 为加密编码方式的安全通信模型下的简易通信流程图。由上图可以发现，该模型使用的物联网终端和物联网 Web 服务器保持不变，仅在通信方式上做了一些处理。为了更好的了解此模型，下面几小节中将分别介绍模型中使用的术语、模型的通信流程、模型的数据传输以及模型的安全性分析。

3.2.2 模型使用的术语

以下是对图 3.2 中用到的几个术语的简单介绍。

设备 ID：与 USERKEY 方式模型下的设备 ID 概念相同，是设备的唯一标识，用于区分不同的物联网终端设备。

原密码：由用户设定的密码，分别保存在物联网终端和物联网 Web 服务器数据库中，在物联网终端获取动态密码阶段使用。

动态密码：由物联网 Web 服务器随机生成，在物联网终端获取动态密码后使用[29]。

计数器：计数器的值由物联网终端负责维护，每向物联网 Web 服务器发送一次请求，计数器值都要加一，并随业务数据一起发送到 Web 服务器，服务器记录对应终端计数器的值，并比较上次计数器的值，如果发现小于或等于上次记录的值则返回错误。计数器机制可以防止攻击者往服务器上重复发送窃听后的数据。

SESSION ID：物联网 Web 服务器为每一个物联网终端建立一个 Session，用于保存物联网终端的设备 ID、动态密码和计数器的值，将这三个常用的值放到 Session 中可以减少数据库的查询次数，减轻数据库服务器的压力。SESSION ID 是服务器找到对应物联网终端 Session 的一个标记。在 HTTP 协议简介小节中提到，HTTP 是一种无状态协议，为了让服务器知道当前请求是上一个客户端发送的请求，常用的两种技术是 Cookie 技术和 Session 技术，而 Session 技术的实现要依赖于 Cookie 技术。在 PHP 中的 Session 技术实现原理是这样的：客户端请求需要设置 Session 的网址时，服务器端会在响应报文的首部行中添加“Set-Cookie: PHPSESSID=6118ff61bea28bb43776de1601ddfb59”字段，客户端需要将该字段保存下来，在下次发起请求时需要将该字段以 Cookie 的方式添加到请求报文的首部行中，例如“Cookie: PHPSESSID=6118ff61bea28bb43776de1601ddfb59”。服务器会根据客户端 Cookie 中 PHPSESSID 字段值找到对应客户端的 Session[30-32]。

业务数据：业务数据是根据用户的实际应用定义的数据，包含从传感器获取到的数据或物联网终端的运行状态数据等等。

原数据：未经加密的数据。使用本模型也可以选用 json 格式来组织数据。

3.2.3 模型的通信流程

上小节是对模型中使用的术语的简单介绍，在此基础上，了解该模型通信流程的就容易多了。该模型中物联网终端和服务器之间的通信流程大致可分为两个阶段：动态密码获取阶段、业务数据上报阶段。下面是对每个阶段的介绍。

动态密码获取阶段：物联网终端上电启动后，首先要向物联网 Web 服务器获取动态密码，为后续的业务数据上报阶段做准备。在该阶段，物联网终端需要上报自己的设备 ID 和使用原密码加密后的计数器数据。服务器收到该请求后，会根据设备 ID 从数据库中找到设备的原密码，并使用原密码解密计数器数据。解密成功后，会创建与该设备对应的 Session，并生成动态密码。在 Session 中保存设备 ID、动态密码、计数器数据。之后，使用原密码加密动态密码，将加密后的得到数据 Base64 编码后返回给物联网终端[33-34]。

由于服务器开启了 Session，所以在响应报文的首部行会将 SESSION ID 返回给物联网终端，它出现在 Set-Cookie 行中。例如在使用 PHP 开发语言时，Set-Cookie 字段定义的 SESSION ID 名称为 PHPSESSID，其值为一个长度为 32 的字符串。物联网终端需要将首部行中的 SESSION ID 保存起来，之后对服务器的每次请求报文中都要将 SESSION ID 放入请求报文的首部行的“Cookie”字段中。

物联网终端收到服务器的响应后，首先 Base64 解码再使用原密码解密，得到动态密码，并保存。

为确保收到的动态密码无误，物联网终端可以发送一次密码确认请求，该请求中包含设备 ID、使用动态密码加密编码后的计数器数据，并在请求报文的首部行中加入 SESSION ID，服务器收到带有 SESSION ID 的请求报文后会根据 SESSION ID 找到对应的 Session，在 Session 中找到该设备的动态密码、以及上次保存的计数器值。服务器根据动态密码解密数据，并判断得到的计数器值是否大于上次保存的值，最后做出回应。例如：成功返回“ok”，失败返回“err”。

业务数据上报阶段：业务数据上报阶段与用户的实际应用息息相关，用户可以自己定义要上报哪些数据、数据的传输格式以及如何处理来自服务器的响应。在该阶段，物联网终端与服务器之间使用动态密码加解密数据。

下面是对通信流程总结：在通信的过程中，物联网终端和物联网 Web 服务器都维护着一些数据。物联网终端维护的数据有设备 ID、原密码、动态密码、计数器、SESSION ID、业务数据。物联网 Web 服务器维护的数据有 Session 数据和数据库数据，Session 数据保存设备 ID、动态密码、计数器。数据库中保存设备 ID、原密码、业务数据。在动态密码获取阶段，通信双方协商好动态密码。在业务数据上报阶段，通信双方使用协商好的动态密码进行加解密数据。

3.2.4 模型的数据传输

该模型的数据传输如图 3.3 所示。

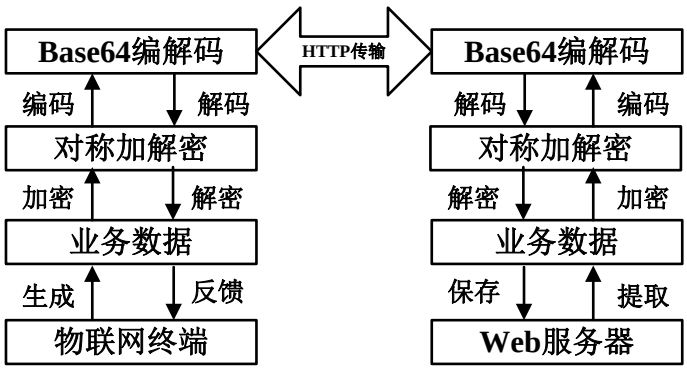

图 3.3 数据传输

Fig. 3.3 Data transmission

针对 HTTP 协议的特点，以及考虑到物联网终端与服务器程序的开发难易程度，该模型的数据传输不直接传输加密后产生的二进制数据，而是将二进制数据 Base64 编码后再进行传输。

如图 3.3 所示，物联网终端根据实际应用产生业务数据，业务数据经过对称加密后，生成二进制数据，二进制数据经过 Base64 编码生成适合 HTTP 传输的密文。Web 服务器收到密文以后，首先进行 Base64 解码，解码后得到加密后的二进制数据，再使用密码对二进制数据进行解密，得到明文的业务数据，服务器根据实际的应用对业务数据进行处理。对处理后生成的明文业务数据按照接收数据的反方向进行处理，先对称加密再 Base64 编码，最后返回给客户端。客户端收到服务器的响应后也按照发送数据的反放向流程进行数据处理，先 Base64 解码再对称解密，最后得到业务数据，进行相应处理。

3.2.5 模型的安全性分析

第 3.1.3 节中提到，基于 USERKEY 的安全通信模型安全性不高，攻击者容易截获、伪造、篡改通信数据，而对于加密编码方式的安全通信模型能否抵御以上几种攻击，以下是对其安全性能的分析。

1. 攻击者截获物联网终端与服务器的通信数据

如果攻击者“窃听”物联网终端与服务器之间的通信数据，由于通信数据是经过对称加密后再 Base64 编码的数据，攻击者仅能对“窃听”后的数据进行 Base64 解码，由于不知道对称加密的密码，而只能用密码枚举法破解数据，攻击者短期内无法破解该数据。

2. 攻击者截获的数据并再次发送到服务器

由于在物联网终端和 Web 服务器通信数据中加入了计数器，因此服务器判断计数器的值即可验证是否是合法数据，从而防止了攻击者的重放攻击[35-36]。

3. 攻击者伪造数据并发送到服务器

攻击者由于没有通信使用的实际密码，因此伪造的数据服务器端无法解密，服务器认为是无效数据。

4. 攻击者篡改数据并发送到服务器

攻击者篡改窃听后数据后，再发送至服务器，服务器端由于无法解密出正确的值而认为是无效数据。

5. 攻击者将物联网终端移走

服务器端可以增加物联网终端心跳检测机制，发现物联网终端在按规定时间内没有新的心跳，则认为终端异常，则触发报警。这种服务器检测心跳机制可以有效保障物联网终端的设备安全。

3.2.6 结论

通过以上几小节的介绍和分析，发现加密编码方式的安全通信模型虽然增加了一些流程，但在总体上没有改变“物联网终端-WEB 服务器”架构，该模型仍然使用 HTTP 协议进行传输，具有开发方便、使用简单的特点。

由于增加了加解密流程，很大程度上增加了物联网终端与服务器的通信安全，可以为将来物联网终端大规模部署奠定基础。

3.3 本章小结

本章首先介绍了 USERKEY 方式的安全通信模型，随后列举国了内三家公司使用该模型的实例，并结合实例对该模型进行分析，发现该模型的安全性不高。随后针对 USERKEY 方式的安全通信模型的安全性的不足，提出了加密编码方式的安全通信模型，并加以深入介绍分析。通过对加密编码方式的安全通信模型的介绍和分析，可以为下一步实现该模型提供指导。

4 加密编码方式的安全通信模型的实现

本章为论文中提出的新理论模型的实现部分。将全方位地介绍如何对第三章提出的理论模型加以实现。本章将先提出一个应用场景，再根据应用场景并结合“物联网终端-WEB 服务器”架构，全方位地介绍如何把加密编码方式的安全通信模型应用于架构中的各个层次。

4.1 应用场景

目前大部分家庭用户上网使用的是 ISP 运营商提供的宽带接入服务，用户使用的路由器往往是自己采购的。当网络不稳定或出现问题，往往都需要 ISP 运营商相关人员现场服务，为此增加了 ISP 运营商的运维成本。为了降低运维成本，ISP 运营商可以免费给用户使用可以由 ISP 运营商远程维护和管理的路由器。一方面，可以降低运营商的服务成本。另一方面可以及时发现并解决网络故障，进而可以提高家庭上网客户对运营商服务的满意度。

要满足应用场景中提出的需求，实现路由器的远程维护和管理至少需要具备以下基本要素：

1. 供运维人员使用的后台管理系统。

2. 管理系统中实现路由器运行的状态查询：包括在线状态、上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等等。

3. 保证路由器与服务器之间数据通信的安全。

传统 USERKEY 方式的安全通信模型并不能满足以上需求，而加密编码方式的安全通信模型恰能弥补其不足。

4.2 模型在物联网终端上的实现

4.2.1 物联网终端的选择

要实现具备主动上报运行信息功能的路由器，要求该路由器软件系统具有可扩展能力，结合应用场景，本文中选用的路由器为支持 Openwrt 系统的路由器。

Openwrt 是一个嵌入式 linux 的发行版，目前被广泛应用于路由器、智能家居、小型机器人等设备中[37-38]。选用支持 Openwrt 的路由器，符合“物联网终端-WEB 服务器”架构中物联网终端的基本条件（见 2.2.1 节），并且 Openwrt 系统是一个开源软件，有丰富的开发包和工具包，它为新功能的扩展提供了有利的条件。

4.2.2 主程序设计

物联网终端的主程序流程图如图 4.1 所示：

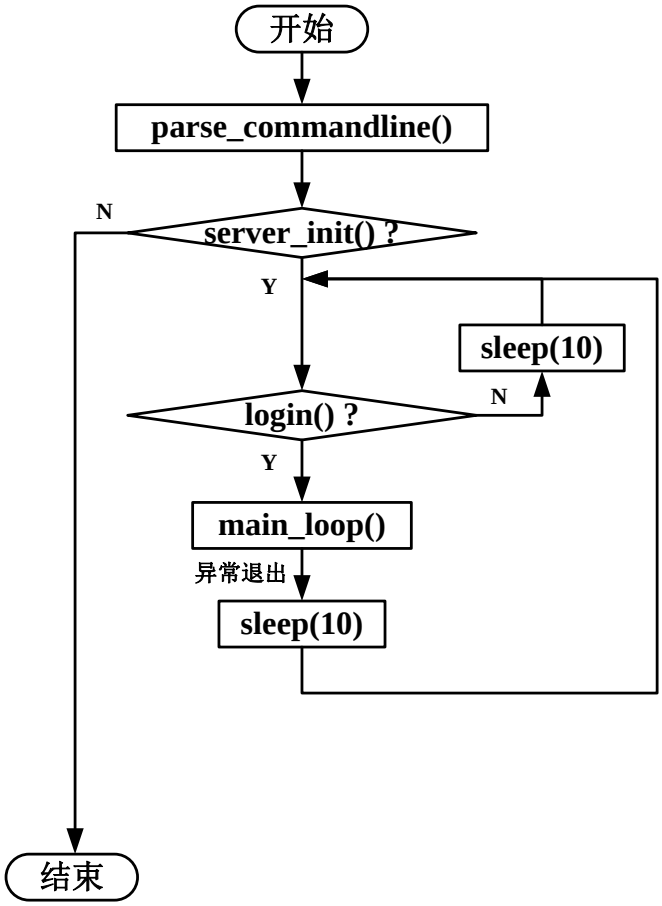

图 4.1 主程序流程图

Fig. 4.1 Main program flow chart

程序开始后，先执行 parse_commandline()函数，该函数用于解析用户传入的命令行参数，用户调用主程序时，可以传入设备 ID、设备原密码、调试打印级别等等。

之后执行 server_init()函数，该函数用于初始化服务器信息，将服务器的域名转化成 IP 地址。如果 server_init()函数执行失败，该程序将直接退出。

login()函数实现了通信流程中的动态密码获取阶段，该函数执行成功程序将直接进入通信流程中的业务数据上报阶段，否则程序将睡眠一段时间后继续尝试动态密码获取。

main_loop()函数是通信流程中的业务数据上报阶段，是与业务息息相关的函数，在该应用需求中，需要每隔一段时间上报路由器的运行状态信息。main_loop()函数如果异常退出，进程将睡眠一段时间后，重新执行动态密码获取阶段，即 login()函数。

下面将分两个小节分别介绍动态密码获取阶段的 login()函数和业务数据上报阶段的 main_loop()函数。

4.2.3 动态密码获取阶段

动态密码获取阶段对应的函数是 login()函数，流程图如图 4.2 所示：

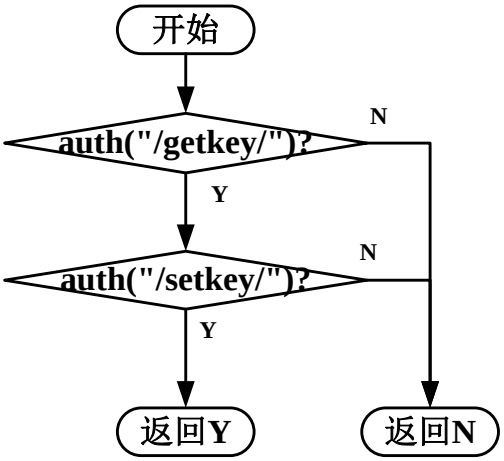

图 4.2 login()函数流程图

Fig. 4.2 Flow chart of login() function

login()函数分为两个阶段，getkey 阶段和 setkey 阶段。

在 getkey 阶段中，进程使用原密码加密，访问物联网 Web 服务器的“/getkey/”路径，获取动态密码并保存，如果获取失败将直接返回 N，并退出函数。获取成功则执行 setkey 阶段函数。

在 setkey 阶段中，进程使用动态密码加密，访问物联网 Web 服务器的“/setkey/”路径，通过 setkey 阶段服务器能确认物联网终端是否获取了正确的密码，如果确认失败将直接返回 N，并退出函数。获取成功程序将返回 Y。

图 4.3 是 auth()函数的流程图。

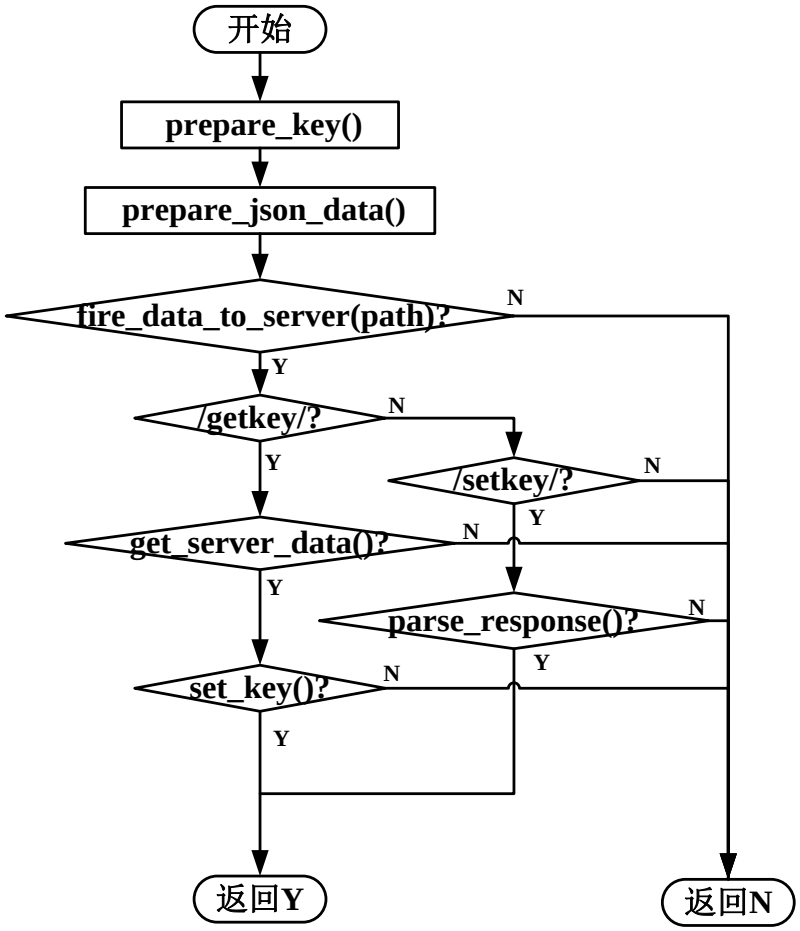

图 4.3 auth()函数流程图

Fig. 4.3 Flow chart of auth() function

auth()函数接收一个参数 path ，path 为请求服务器的路径。

auth()函数首先执行 prepare_key()函数准备通信密码。prepare_key()函数会判断是 getkey 阶段还是 setkey 阶段，如果是 getkey 阶段则使用原密码，如果是 setkey 阶段则使用动态密码。

prepare_json_data()函数，用于准备 json 格式的数据，无论是在 getkey 阶段还是 setkey 阶段，json 格式的数据中都要包含计数器，并且计数器的值是累加的，其他的数据是用户的业务数据。

fire_data_to_server(path)函数，请求路径为 path，将加密编码后的 json 格式的数据发送到服务器。如果 fire_data_to_server()函数失败，则直接返回 N，如果成功，服务器的响应报文就存到了全局数组 response_buf 中。

继续往下执行，判断如果是 getkey 阶段，则执行 get_server_data()函数，get_server_data()函数会将服务器的响应报文中的实体主体部分先 Base64 解码后使用原

密码进行解密，会得到含有动态密码的 json 格式的数据。之后交给 set_key()函数处理，set_key()函数负责解析含有动态密码的 json 格式数据，并将动态密码保存。

如果是 setkey 阶段，则执行 parse_response()函数。parse_response()函数直接判断服务器的响应报文中的实体主体部分是不是“ok”字符串，如果是，则表明动态密码确认成功，如果不是则直接返回 N。

图 4.4 是 fire_data_to_server()函数的流程图。

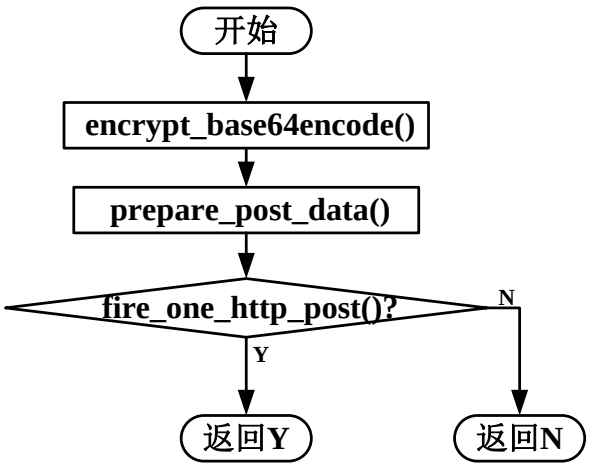

图 4.4 fire_data_to_server()函数流程图
Fig. 4.4 Flow chart of fire_data_to_server() function

fire_data_to_server()函数接收三个参数：path、str_json、key，其中 path 为请求服务器的路径，str_json 为传入的 json 字符串，key 为加密使用的密码。

该函数先使用 encrypt_base64encode 函数，将传入的 str_json 字符串使用 key 加密，之后再进行编码，生成数据 data。

之后，使用 prepare_post_data()函数组装将要 POST 的数据，组装格式为“id=XXX&&data=XXX”，组装完毕以后，使用 fire_one_http_post()函数将组装数据发送到物联网 Web 服务器。

图 4.5 是 fire_one_http_post()函数的流程图。

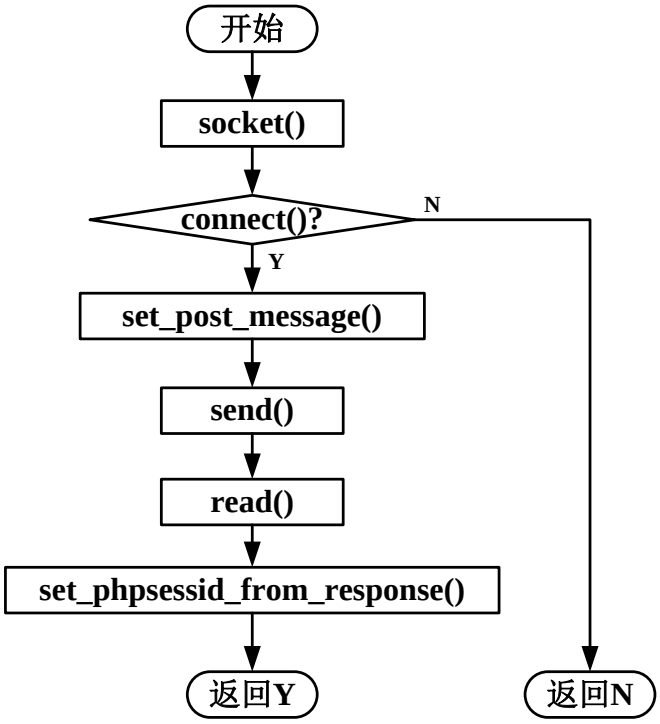

图 4.5 fire_one_http_post()函数流程图
Fig. 4.5 Flow chart of fire_one_http_post() function

fire_one_http_post()函数接收两个参数：path、content，其中 path 为请求服务器的路径，content 为 POST 的实体主体。

该函数先执行 socket()函数，获取套接字 fd，再执行 connect()函数，连接至物联网 Web 服务器，如果连接失败直接返回 N。如果连接成功，则使用 set_post_message()函数组装 POST 请求报文。组装完毕以后，使用 send()函数将报文发送到服务器，紧接着使用 read()函数接收来自服务器的响应报文。

set_phpsessid_from_response()函数将从服务器的响应报文的首部行中分析有无“Set-Cookie: PHPSESSID=xxxxxx”，并将找到的 PHPSESSID 保存，以备下次发送请求报文时使用。

4.2.4 业务数据上报阶段

业务数据上报阶段对应的是 main_loop()函数，它的流程图如图 4.6 所示，

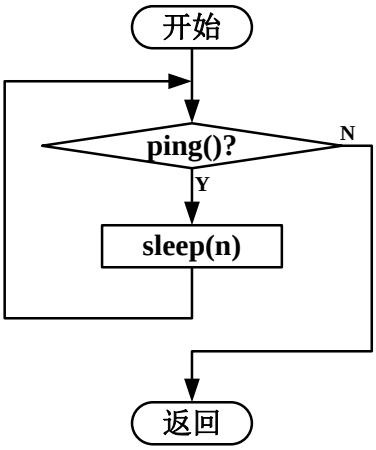

图 4.6 main_loop()函数流程图
Fig. 4.6 Flow chart of main_loop() function

main_loop()函数每隔一段时间执行一次 ping()函数，如果 ping()函数出错，则跳出循环返回。ping()函数是与业务相关的函数，在该应用中，它负责收集路由器的运行信息，并上报给物联网 Web 服务器。

ping()函数的流程图如图 4.7 所示。

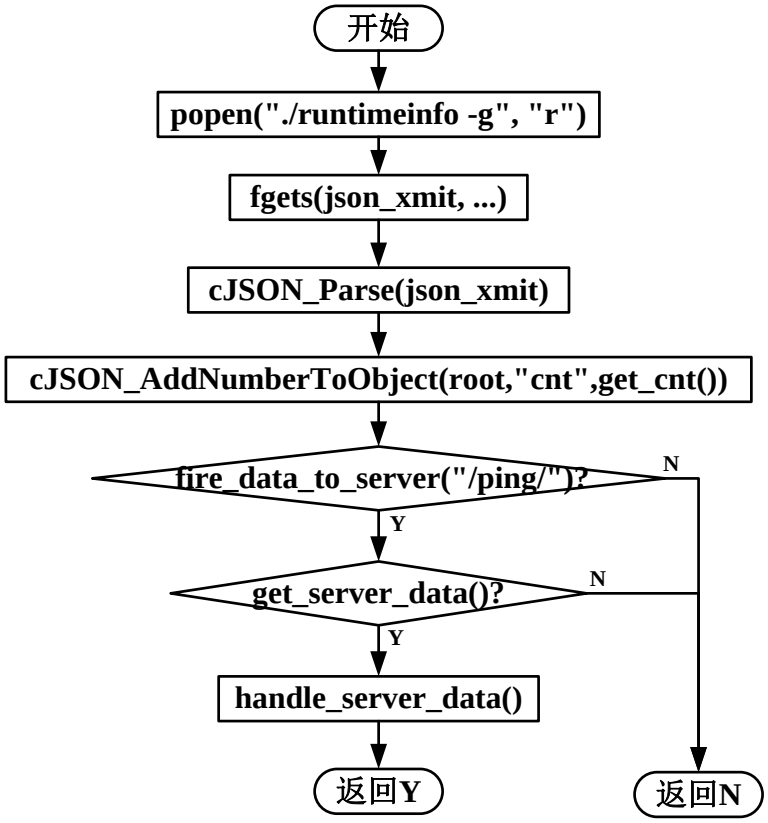

图 4.7 ping()函数流程图
Fig. 4.7 Flow chart of ping() function

ping()函数使用 popen()函数调用“runtimeinfo”脚本。“runtimeinfo”脚本用于获取路由器的上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等信息，并能按照 json 格式的数据输出。popen()函数调用的脚本执行结束后，使用 fgets()函数读取脚本返回后的结果。

由于模型中规定用户的业务数据中要包含计数器值，因此接下来使用了 cJSON_Parse()函数和 cJSON_AddNumberToObject(root,"cnt",get_cnt())函数，把“runtimeinfo”脚本返回的数据中加入了计数器字段。例如，加入计数器后，json 格式的数据为“{"opIdentification":"runtimeinfo-status","uptime":"344247","loadAverage":"0.00, 0.03, 0.04","memory":"38188/61188","client":"2","trafficWan":"2845/27357","trafficLan":"322/130","trafficWifi":"34695/3488","proto":"dhcp","cnt":3}”。

fire_data_to_server(“/ping/”)函数，请求路径为“/ping/”，把使用动态密码加密编码后的 json 格式的数据发送到服务器。get_server_data()函数将服务器返回的数据解码解密，送给 handle_server_data()函数。

4.3 模型在 WEB 服务器上的实现

4.3.1 Web 服务器的选择

本设计选用新浪提供的 SAE 作为 Web 服务器。SAE 全称为 Sina App Engine，中文翻译为新浪云应用，它是国内最具影响力的，分布式 Web 应用/业务开发托管、运行平台。

SAE 使用 PaaS 的方式，具有计费粒度更细、高可靠、高扩展、免运维等特点，它非常适合 HTTP 业务[39-41]。

4.3.2 数据库设计

根据 4.1 节提出的实际应用需求，数据库至少要包括两类数据表格，一类是面向物联网终端的数据表，一类是面向用户的数据表。

面向物联网终端的数据表要包含物联网终端的设备 ID、设备原密码和其他业务相关的数据。针对该实际应用需求的业务相关数据为：路由器的上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等。

向用户的数据表要包含用户 ID、用户名、用户密码、注册时间等。

面向物联网终端的数据表表名为 nodes，其表结构以及各个字段的说明如表 4.1 所示。

表 4.1 nodes 表
Table 4.1 nodes

字段	类型	属性	说明
id	int(10)	UNSIGNED	设备 ID
uid	int(10)	UNSIGNED	用户 ID
name	varchar(150)	-	设备名称
passwd	varchar(50)	-	设备密码
created_at	int(10)	UNSIGNED	设备创建时间
last_heartbeat_at	int(10)	UNSIGNED	设备最新心跳时间
uptime	int(10)	UNSIGNED	设备运行时间
loadAverage	varchar(20)	-	设备负载
memory	varchar(20)	-	设备内存
client	int(10)	UNSIGNED	设备接入用户数量
trafficWan	varchar(30)	-	设备 WAN 口网速
trafficLan	varchar(30)	-	设备 LAN 口网速
trafficWifi	varchar(30)	-	设备 WiFi 网速
proto	varchar(10)	-	设备上网协议

面向用户的数据表表名为 user_info，其表结构以及各个字段的说明如表 4.2 所示。

表 4.2 user_info 表
Table 4.2 user_info

字段	类型	属性	说明
id	int(10)	UNSIGNED	用户 ID
name	varchar(30)	-	用户名
passwd	varchar(32)	-	用户密码
regtime	int(10)	UNSIGNED	用户注册时间

4.3.3 动态密码获取阶段

由 4.2.3 节知道，路由器动态密码获取阶段使用的是 login()函数。该函数的执行分为两个阶段，分别是 getkey 阶段和 setkey 阶段，对应访问服务器的路径分别是“/getkey/”和“/setkey/”，因此在服务器端要实现这两个路径的脚本。例如，本文使用的是 PHP 开发语言实现，因此，要在服务器的根目录下分别创建两个 PHP 脚本，分别是“/getkey/index.php”和“/setkey/index.php”。以下是这两个脚本的处理流程。

1. getkey 阶段

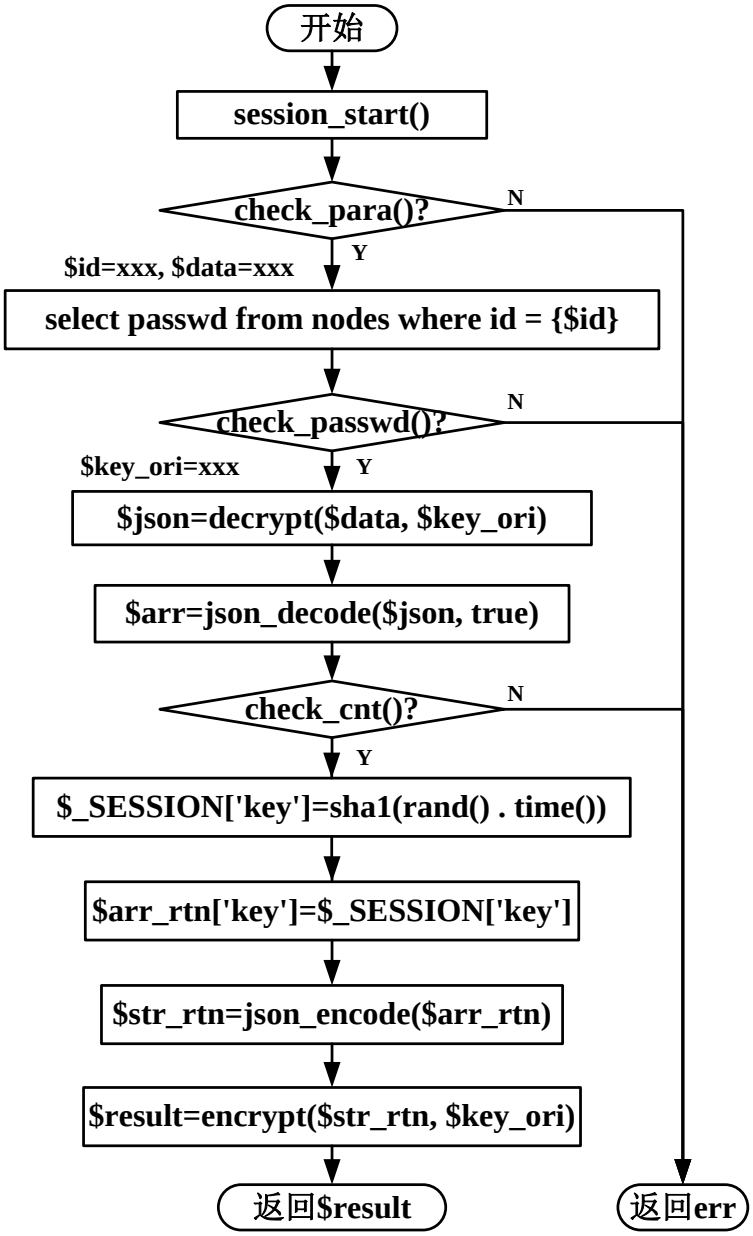

图 4.8 getkey 阶段流程图

Fig. 4.8 Flow chart of getkey stage

getkey 阶段处理流程如图 4.8 所示，在 getkey 阶段，服务器收到来自于物联网终端的请求之后，首先执行 session_start()函数，开启 Session 功能，为以后保存 Session 数据做准备。

之后，执行 check_para()函数，该函数判断物联网终端的请求报文中是不是同时具有 id 和 data 两个参数，如果参数不全，则返回 err 并立即退出。如果参数齐全，则将 id 和 data 赋值给$id 和$data 变量。

继续执行数据库查询，根据$id 来查询对应物联网终端的原密码。

check_passwd()函数检测查询数据库后的结果，如果数据库中没有对应$id 的 passwd，则返回 err 并立即退出。如果存在 passwd 则将 passwd 赋值给$key_ori 变量。

之后，使用$key_ori 和 decrypt()函数 Base64 解码并解密$data，将解码解密后的数据赋值给$json 变量。

使用 json_decode()函数解析$json 的值。得到关联数组$arr。

check_cnt()函数判断关联数组$arr 中是否含有$arr[‘cnt’]，如果没有，则代表物联网终端没有上传计数器或者是攻击者伪造的报文，在这种情况下，返回 err 并立即退出。如果存在$arr[‘cnt’]，则表示解密成功。check_cnt()函数会继续检测 Session 中有没有保存过计数器值。如果没有，表示该终端上电后首次登录服务器，该函数会记录该计数器的值到 Session 中。如果有，表示该终端上电后连过服务器，由于某种原因重新登录，这时，函数会比较 Session 中保存的计数器值和终端上报的计数器值，如果终端上报的计数器值大于 Session 中保存的计数器值则表示该请求报文有效并将 Session 中保存的计数器值更新为终端刚上报上来的计数器的值，否则认为是攻击者的重发报文攻击，在这种情况下，返回 err 并立即退出。

对计数器的值检测正常后，服务器会使用哈希算法生成一个动态密码，并保存到 Session 中。

之后，将动态密码放入一个关联数组中，并使用 json_encode()函数将关联数组转换成 json 格式的数据赋值给$str_rtn。

最后使用$key_ori 和 encrypt()函数将$str_rtn 加密并 Base64 编码返回给物联网终端。

2. setkey 阶段

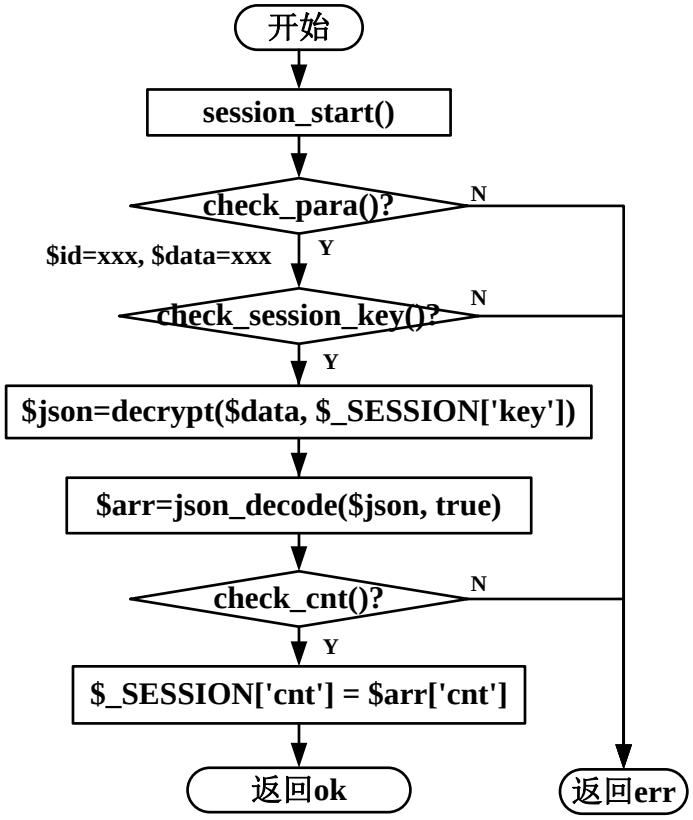

图 4.9 setkey 阶段流程图

Fig. 4.9 Flow chart of setkey stage

setkey 阶段处理流程如图 4.9 所示，在 setkey 阶段，服务器收到来自于物联网终端的请求之后，也是首先执行 session_start()函数，开启 Session 功能，用于读取之前保存的 Session 数据。

之后，执行 check_para()函数，该函数判断物联网终端的请求报文中是不是同时具有 id 和 data 两个参数，如果参数不全，则返回 err 并立即退出。如果参数齐全，则将 id 和 data 赋值给$id 和$data 变量。

按照获取动态密码阶段的流程，在 getkey 阶段，服务器已经生成了动态密码并存入了 Session 中。执行 check_session_key()函数，会检查服务器对应该物联网终端的 Session 中有没有动态密码。如果没有，表明物联网终端的获取动态密码的流程错误，或者是来自攻击者的攻击，在这种情况下，返回 err 并立即退出。

如果 Session 中有动态密码，则使用动态密码和 decrypt()函数解码并解密$data 数据，得到的结果存入$json 中。

使用 json_decode()函数解析$json 的值。得到关联数组$arr。

check_cnt()函数功能与之前介绍的功能一致，在这不做介绍。

对计数器的值检测正常后，Session 中保存的计数器值会更新为终端刚上报上来的计数器的值，之后服务器返回 ok。

4.3.4 业务数据上报阶段

由 4.2.4 节知道，路由器业务数据上报阶段使用的是 main_loop()函数，该函数的执行是一个循环，每隔一段时间向服务器上报一次业务数据，访问服务器的路径是 “/ping/”，因此在服务器端要实现这个路径的脚本。本文使用的是 PHP 开发语言实现，因此，要在服务器的根目录下创建 PHP 脚本 “/ping/index.php”。以下是这个脚本的处理流程。

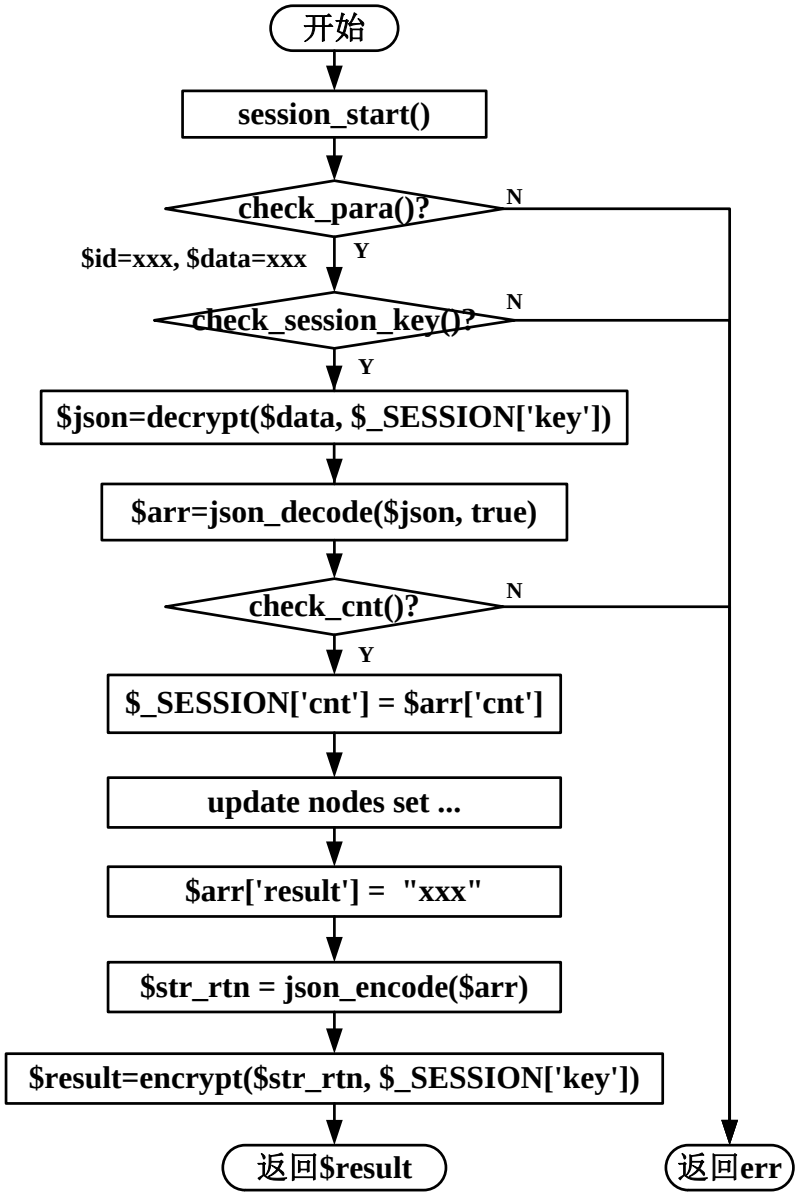

图 4.10 ping 阶段流程图

Fig. 4.10 Flow chart of ping stage

ping 阶段处理流程如图 4.10 所示，在该阶段，服务器收到来自于物联网终端的请求之后，也是首先执行 session_start()函数，开启 Session 功能，用于读取之前保存的 Session 数据。

之后，执行 check_para()函数，该函数判断物联网终端的请求报文中是不是同时具有 id 和 data 两个参数，如果参数不全，则返回 err 并立即退出。如果参数齐全，则将 id 和 data 赋值给$id 和$data 变量。

check_session_key()函数，会检查服务器对应该物联网终端的 Session 中有没有动态密码。如果没有，表明物联网终端的获取动态密码的流程错误，或者是来自攻击者的攻击，在这种情况下，返回 err 并立即退出。

如果 Session 中有动态密码，则使用动态密码和 decrypt()函数解码并解密$data 数据，得到的结果存入$json 中。

使用 json_decode()函数解析$json 的值。得到关联数组$arr。

check_cnt()函数功能与之前介绍的功能一致，在这不做介绍。

对计数器的值检测正常后，Session 中保存的计数器值会更新为终端刚上报上来的计数器的值。

根据业务流程，将物联网终端上报上来的其他数据保存到数据库中。

4.4 物联网终端管理系统的实现

物联网终端管理系统是搭建在 Web 服务器上的、面向用户使用的系统。用户可以使用手机、平板或者电脑等任意一个具有 Web 浏览器的终端来访问该管理系统。用户使用该管理系统首先需要在该系统上注册一个帐号，之后才能添加自己的设备。当用户的设备上线以后，用户可以通过该系统查看自己设备当前的运行状态以及远程控制这些设备。

物联网终端管理系统包括用户管理模块和设备管理模块，分别用于管理用户和设备。下面小节中将简单介绍这两个模块的实现方法。

4.4.1 用户管理模块的实现

用户管理模块主要实现用户的注册、登录、退出、修改密码等功能。

用户的注册和登录放到系统首页，用户输入物联网终端管理系统的域名以后，即可显示登录和注册界面。登录和注册界面如图 4.11 所示。

图 4.11 登录和注册界面

Fig. 4.11 Interface of login and registration

图 4.11 为用户的登录注册界面，对于已注册过的用户，输入用户名和密码后单击登录即可。对于新用户，需要单击用户注册，会弹出用户注册对话框，如图 4.12 所示。

图 4.12 用户注册对话框

Fig. 4.12 User registration dialog

图 4.12 为用户注册对话框，新用户注册时需要填写用户名、密码、确认密码信息，填写无误后单击提交注册即可。

以下是注册与登录流程图。

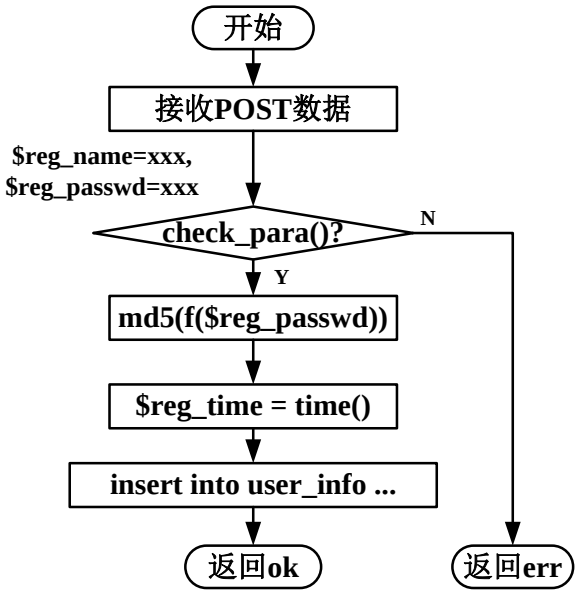

图 4.13 用户注册流程图

Fig. 4.13 Flow chart of user registration

用户单击提交注册按钮之后，浏览器中的 JavaScript 脚本[42-44]会向服务器发起 Ajax 请求[45-47]，将用户输入的数据以 POST 的方式发送到服务器，图 4.13 的流程图是服务器对 Ajax 请求的处理流程。首先接收 POST 数据，之后执行 check_para()函数，该函数检测用户输入的数据是否合法，包括有无非法字符、sql 注入语句、用户名是否重复等等。如果用户输入的数据不合法，则返回 err 并立即退出。如果合法，则对用户输入的密码进行处理，服务器将保存处理后的数据。之后，获取系统的当前时间作为用户的注册时间。最后执行 SQL 语句将用户的注册信息写入数据库并返回 ok。JavaScript 脚本会根据服务器返回的数据提示用户注册成功或者失败。

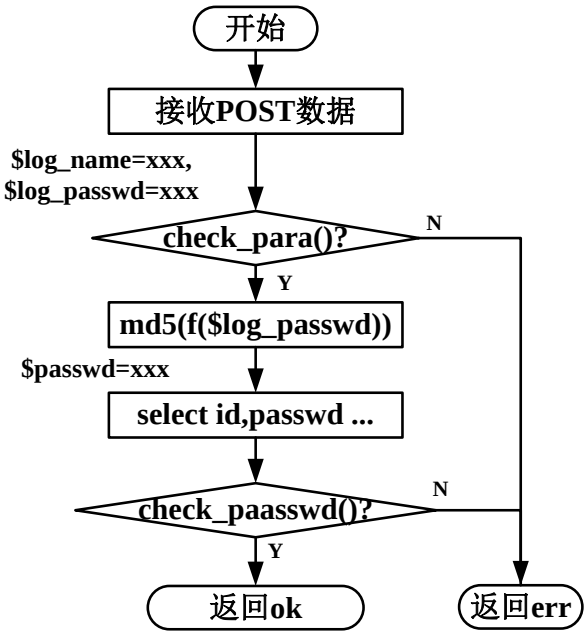

图 4.14 用户登录流程图

Fig. 4.14 Flow chart of user login

在登录界面，用户输入用户名和密码并单击登录按钮后，浏览器中的 JavaScript 脚本会向服务器发起 Ajax 请求，将用户输入的数据以 POST 的方式发送到服务器，图 4.14 的流程图是服务器对 Ajax 请求的处理流程。首先接收 POST 数据，之后执行 check_para() 函数，该函数检测用户输入的数据是否合法，包括有无非法字符、sql 注入语句、用户名是否存在等等。如果用户输入的数据不合法，则返回 err 并立即退出。如果合法，则对用户输入的密码进行处理。之后，执行 SQL 语句将用户的注册时的密码从数据库中取出来。使用 check_passwd()函数判断密码的正确性。如果密码正确则返回 ok，否则返回 err。JavaScript 脚本会根据服务器返回的数据提示用户登录成功或者失败。

当用户需要修改密码时，需要登录到系统，找到修改密码界面。如图 4.15 所示。

图 4.15 修改密码界面

Fig. 4.15 Interface of modifing password

在修改密码界面，用户输入原密码、新密码和确认密码并单击提交按钮后，浏览器中的 JavaScript 脚本会向服务器发起 Ajax 请求，将用户输入数据以 POST 的方式发送到服务器。

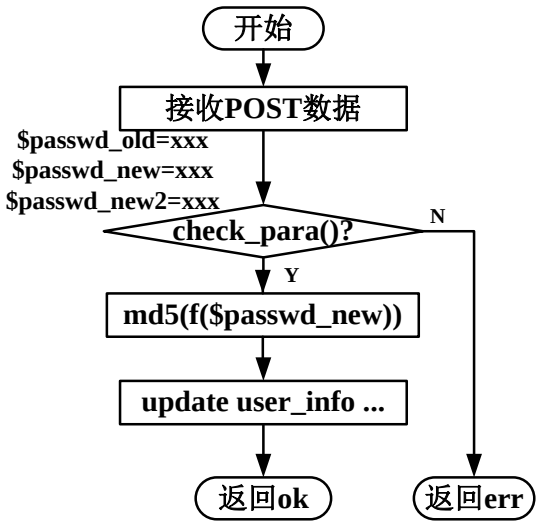

图 4.16 修改密码流程

Fig. 4.16 Flow chart of modifing password

图 4.16 的流程图是服务器对 Ajax 请求的处理流程。首先接收 POST 数据，之后执行 check_para()函数，该函数检测用户输入的数据是否合法，包括有无非法字符、sql 注入语句、原密码是否正确、新密码是否正确等等。如果用户输入的数据不合法，则返回 err 并立即退出。如果合法，则对用户输入的新密码进行处理。最后执行 SQL 语句将用

户的新密码写入数据库并返回 ok。JavaScript 脚本会根据服务器返回的数据提示用户修改密码成功或者失败。

4.4.2 设备管理模块的实现

设备管理模块实现设备的查看、添加、配置、删除等功能。由于设备的查看、添加、配置、删除等功能的实现均使用浏览器中的 JavaScript 脚本执行 Ajax 请求，其与服务器交互流程与 4.4.1 节中用户的注册、修改密码流程类似，因此本节将不再赘述。

当用户使用新注册后的帐号登录到系统时，对应帐号下没有设备，用户需要首先添加新设备。系统的主界面如图 4.17 所示。

图 4.17 主界面

Fig. 4.17 Main interface

用户单击添加设备按钮，就会弹出添加新设备对话框，如图 4.18 所示。新设备添加时需要填写设备名、设备密码（3.2.2 节中提出的原密码），确认填写无误后单击提交即可。

图 4.18 添加新设备对话框

Fig. 4.18 Dialog of adding new device

假设用户添加的新设备名为“测试路由器 1”，设备密码为“testforpaper”，则单击提交后的效果如图 4.19 所示。

图 4.19 设备展示

Fig. 4.19 List of devices

由于“测试路由器 1”还未接入网络，还没有上报过信息，因此图 4.19 展示的路由器的各种运行信息为空，并且系统显示设备未上线。

单击查看按钮将弹出设备详细信息对话框。通过该对话框可以查看设备的详细信息，如图 4.20 所示。设备的详细信息中展示了设备的管理者、设备名称、设备 ID、设备密码以及设备添加时间。

图 4.20 设备详细信息对话框
Fig. 4.20 Dialog of device detail

单击配置按钮将弹出设备配置对话框。通过该对话框可以配置设备，如图4.21所示。设备的配置对话框中展示了设备的管理者、设备名称、设备密码。用户可以修改设备名称以及设备密码。

图 4.21 配置设备对话框
Fig. 4.21Dialog of device configure

单击删除按钮将弹出设备删除对话框。如图 4.22 所示。设备删除对话框中展示了删除设备的警告信息。用户单击确定后，设备将被从管理系统中删除。

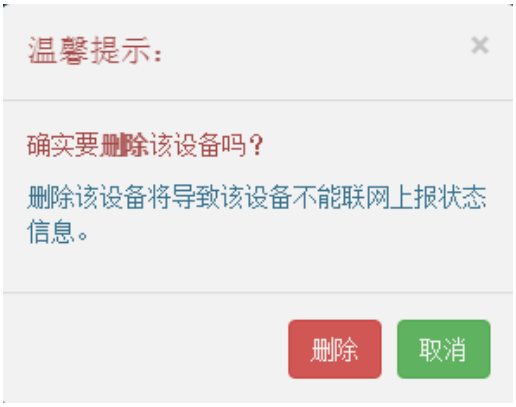

图 4.22 配置删除对话框
Fig. 4.22 Dialog of removing device

当“测试路由器 1”对应的物联网设备接入网络并向物联网 Web 服务器上报信息后，在管理系统中就能看到对应的信息，如图 4.23 所示。图中显示了路由器的运行状态，包括在线状态、上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等等。此时，设备运行正常，能正常联网。

图 4.23 设备展示
Fig. 4.23 List of devices

当设备出现异常，不能正常联网时，管理系统中会显示该设备不在线，如图 4.24 所示。“更新时间”一栏中，显示了设备最后一次上报数据的时间以及异常发生时的上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等等。

图 4.24 设备展示
Fig. 4.24 List of devices

4.5 本章小结

本章先提出了一个应用场景，之后根据应用场景并结合“物联网终端-WEB 服务器”架构，全方位地介绍了如何将加密编码方式的安全通信模型应用于该架构中的各个层次。

本章首先讲述了该模型如何在物联网终端上实现，其对应的是架构中的物联网终端层，其次讲述了该模型如何在 Web 服务器上实现，其对应的是架构中的服务器层，最后讲述了如何实现物联网终端管理系统，其对应的是架构中的用户应用层。

5 加密编码方式的安全通信模型的测试

本章为论文中提出的新理论模型的测试部分。将对第四章中介绍的实现结果进行测试，包括实现程序的使用方法、以及模拟了各种攻击来验证新理论模型的安全性。

5.1 程序说明

经过交叉编译后的物联网终端程序命名为 client，执行 client -h 命令即可显示该终端程序的使用方法。

```
Usage: client [options]

  -h              Print usage
  -v              Print version information
  -i <client_id>    set client_id
  -k <key>    set client key
  -s <sec>    set sleep second [1-1000]
  -d <level>    Debug level
  level:
    0 -LOG_EMERG
    1 -LOG_ALERT
    2 -LOG_CRIT
    3 -LOG_ERR
    4 -LOG_WARNING
    5 -LOG_NOTICE
    6 -LOG_INFO
    7 -LOG_DEBUG
```

图 5.1 client 命令的使用方法

Fig. 5.1 Usage of client commamd

图 5.1 展示了 client 命令的使用方法，执行命令加上-h 选项，将打印使用方法。执行命令加上-v 选项，将打印程序的版本信息。执行命令加上-d 选项，设置程序运行时的打印级别，程序支持 0-7 级打印级别。-i 选项指定该物联网终端的设备 ID，-k 选项指定该物联网终端的设备原密码，-s 选项指定该物联网终端多久上报一次数据。-i 选项、-k 选项为必填选项，它是区别物联网终端和保证物联网终端与服务器正常通信的唯一凭证。

图 5.2 是执行 client -i 30 -k testforpaper -d7 后的部分打印信息。

```
[INFO][Thu Mar 17 15:38:01 2016][21738](client.c:38) client_id: 30
[INFO][Thu Mar 17 15:38:01 2016][21738](client.c:39) client key: testforpaper
[INFO][Thu Mar 17 15:38:01 2016][21738](client.c:40) sleep: 1
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:175) official name of host: iotcloud4test.applinz
i.com
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:176) IP Address :202.108.35.245
[INFO][Thu Mar 17 15:38:01 2016][21738](client.c:51) login start...
```

图 5.2 程序打印信息

Fig. 5.2 Information from program log

从图中可以看出，程序接收的设备 ID 和设备原密码分别是 30 和 testforpaper，默认上报数据间隔为 1 秒。程序在执行初始化阶段解析域名“iotcloud4test.applinzi.com”得

到对应的IP地址为“202.108.35.245”，初始化成功后执行login()函数，打印出“login start...”，接下来是获取动态密码阶段和业务数据上报阶段，将在下节中介绍。

5.2 通信测试

5.1.1 正常通信

程序执行完初始化阶段后，首先要执行动态密码获取阶段。按照4.2.3节的介绍，程序先访问服务器的“/getkey/”路径来获取动态密码，后访问“/setkey/”路径来确认动态密码获取是否正确。图5.3是程序在getkey阶段的打印信息。

```
[INFO][Thu Mar 17 15:38:01 2016][21738](client.c:51) login start...
[INFO][Thu Mar 17 15:38:01 2016][21738](business.c:38) cJSON_CreateObject: {"cnt":1}
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:20) len_buf_encrypt_base64encode: 13
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:44) encrypt_base64encode in
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:54) encrypt_base64encode out
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:121) data_encrypt_base64: D0cQGhJNSEEc
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:126) data_post: id=30&&data=D0cQGhJNSEEc
[INFO][Thu Mar 17 15:38:01 2016][21738](http_post.c:208) connect success
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:130) pos_snprint: 193
[INFO][Thu Mar 17 15:38:01 2016][21738](http_post.c:219) HTTP Request to Server:
-------------
POST /getkey/ HTTP/1.0
Host:iotcloud4test.applinzi.com
Accept:text/html
Accept-Encoding:deflate
Content-Type:application/x-www-form-urlencoded
Content-Length:24

id=30&&data=D0cQGhJNSEEc
-------------
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:247) Read 412 bytes, total now 412
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:75) ssid: 5c21e0396e001c0f926212ef52ca156a
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:153) data_response: D0cYER9NSFJWQgFBFlQWRVBdSk
JRRwQTEgBEQgdaF0UAQgBDEFMRFwRbQUMHQAQUVhg=
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:72) len_base64: 68
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:83) len_add: 1
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:87) len_malloc: 51
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:167) json_server: {"key":"72d3b1e1628207aafe76
a5e5a2e1d6bcb433f0af"}
[INFO][Thu Mar 17 15:38:01 2016][21738](business.c:58) get_server_data: {"key":"72d3b1e1628207aafe76a5
e5a2e1d6bcb433f0af"}
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:88) key: 72d3b1e1628207aafe76a5e5a2e1d6bcb433f
0af
```

图5.3 程序getkey阶段的打印信息

Fig. 5.3 Program log of geykey stage

由图中的打印信息可以看出，程序先准备了计数器数值，并组成为json格式的数据“{"cnt":1}”，接下来将json格式的数据使用原密码加密并Base64编码，生成的数据为“D0cQGhJNSEEc”，之后组装为POST数据“id=30&&data=D0cQGhJNSEEc”。组装成功后，连接服务器，在图中的打印信息可以看到打印的连接成功信息“connect success”。连接成功后，将准备好的POST数据组装到HTTP请求报文中，并将其发送到服务器。请求报文中请求行中对应的URL为“/getkey/”。

服务器收到请求报文后，给予回应。程序分析服务器的响应报文，得到PHPSESSID字段，图中的打印信息为“ssid: 5c21e0396e001c0f926212ef52ca156a”。响应报文的实体主体为加密编码后的数据，图中得到的实体主体的数据为“D0cYER9NSFJWQgFBFlQWRVBdSkJRRwQTEgBEQgdaF0UAQgBDEFMRFwRbQUMHQAQUVhg=”，经过Base6

4 解码，并使用原密码解密后，得到 json 格式的数据“{"key":"72d3b1e1628207aafe76a5e5a2e1d6bcb433f0af"}”，其中的 key 字段对应的值就是从服务器那里获取的动态密码。

接下来是 setkey 阶段，打印信息如图 5.4 所示。

```
[INFO][Thu Mar 17 15:38:01 2016][21738](business.c:38) cJSON_CreateObject: {"cnt":2}
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:20) len_buf_encrypt_base64encode: 13
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:44) encrypt_base64encode in
[DEBUG][Thu Mar 17 15:38:01 2016][21738](crypt_base64.c:54) encrypt_base64encode out
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:121) data_encrypt_base64: TBAHXRYTXwNL
[DEBUG][Thu Mar 17 15:38:01 2016][21738](data_prepare.c:126) data_post: id=30&&data=TBAHXRYTXwNL
[INFO][Thu Mar 17 15:38:01 2016][21738](http_post.c:208) connect success
[DEBUG][Thu Mar 17 15:38:01 2016][21738](http_post.c:130) pos_snprint: 244
[INFO][Thu Mar 17 15:38:01 2016][21738](http_post.c:219) HTTP Request to Server:
-------------
POST /setkey/ HTTP/1.0
Host:iotcloud4test.applinzi.com
Accept:text/html
Accept-Encoding:deflate
Cookie:PHPSESSID=5c21e0396e001c0f926212ef52ca156a
Content-Type:application/x-www-form-urlencoded
Content-Length:24

id=30&&data=TBAHXRYTXwNL
-------------
[DEBUG][Thu Mar 17 15:38:02 2016][21738](http_post.c:247) Read 273 bytes, total now 273
[DEBUG][Thu Mar 17 15:38:02 2016][21738](http_post.c:82) no PHPSESSID=
[DEBUG][Thu Mar 17 15:38:02 2016][21738](business.c:74) data_response: ok
[INFO][Thu Mar 17 15:38:02 2016][21738](client.c:58) key: 72d3b1e1628207aafe76a5e5a2e1d6bcb433f0af
[INFO][Thu Mar 17 15:38:02 2016][21738](client.c:59) login success, trap in main_loop ...
```

图 5.4 程序 setkey 阶段的打印信息

Fig. 5.4 Program log of setkey stage

同样地，由图中的打印信息可以看出，在 setkey 阶段，程序也首先准备了计数器数值，并组成为 json 格式的数据“{"cnt":2}”，这次计数器的值要比上次的值大。接下来将 json 格式的数据使用上次获取到的动态密码加密并 Base64 编码，生成的数据为“TBAHXRYTXwNL”，之后组装为 POST 数据“id=30&&data=TBAHXRYTXwNL”。组装成功后，连接服务器，在图中可以看到打印的连接成功信息“connect success”。连接成功后，将准备好的 POST 数据组装到 HTTP 请求报文中，并将其发送到服务器。请求报文中请求行中对应的 URL 为“/setkey/”。特别地，在请求报文的首部行中加入了 Cookie，对应的字符串为“Cookie:PHPSESSID=5c21e0396e001c0f926212ef52ca156a”，可以看出 PHPSESSID 的值即是上次请求获取到的。

服务器收到请求报文后，给予回应。由图中的打印信息可以看出，服务器回复的数据为“ok”。表明 setkey 阶段执行成功。

程序确认 setkey 阶段执行成功后，打印“login success, trap in main_loop ...”进入业务数据上报阶段。

业务数据上报阶段（即 ping 阶段）的打印信息如图 5.5 所示。

图 5.5 程序 ping 阶段的打印信息

Fig. 5.5 Program log of ping stage

同样地，由图中的打印信息可以看出，在 ping 阶段，程序首先根据业务的实际需要，获取物联网终端的运行状态组成 json 格式的数据，数据中要包含计数器的值。图中组成的 json 格式的数据为“{"opIdentification":"runtimeinfo-status","uptime":"598497","loadAverage":"0.24, 0.30, 0.38","memory":"43040/61188","client":"1","trafficWan":"207/279","trafficLan":"454/164","trafficWifi":"0/10","proto":"dhcp","cnt":3}”，里面包含了应用场景中要求的上网方式、运行时间、负载、内存、接入用户数量、WAN 口网速、LAN 口网速、WiFi 网速等信息和计数器的值，其中计数器的值要比上次的值大。

接下来将 json 格式的数据使用获取到的动态密码加密并 Base64 编码，生成的数据为“TBALQytVAF9...”，之后组装为 POST 数据“id=30&&data=TBALQytVAF9...”。

组装成功后，连接服务器，在图中可以看到打印的连接成功信息“connect success”。连接成功后，将准备好的 POST 数据组装到 HTTP 请求报文中，并将其发送到服务器。请求报文中请求行中对应的 URL 为“/ping/”。特别地，在请求报文的首部行中也加入了 Cookie。

服务器收到请求报文后，给予回应。服务器回复的数据也是使用协商好的动态密码加密并 Base64 编码的数据。物联网终端解码解密数据后做出进一步的响应。

以上是对正常通信流程的详细说明，在实际应用中，很有可能遇到攻击者的攻击，下面小节中模拟了几种攻击，来验证该模型的安全性。

5.1.2 截获攻击

攻击者通过某种手段可以截获物联网终端与服务器的通信数据。通过 5.1.1 节中描述的正常通信流程可以发现，即使攻击者窃听了通信数据，由于没有密码而无法解密通信数据。

5.1.3 重放攻击

攻击者将截获后的数据再次发送到服务器。假设攻击者截获了物联网终端 setkey 阶段的通信报文，并使用工具将截获后的报文重新发给服务器。图 5.6 展示了重放攻击服务器的返回结果。

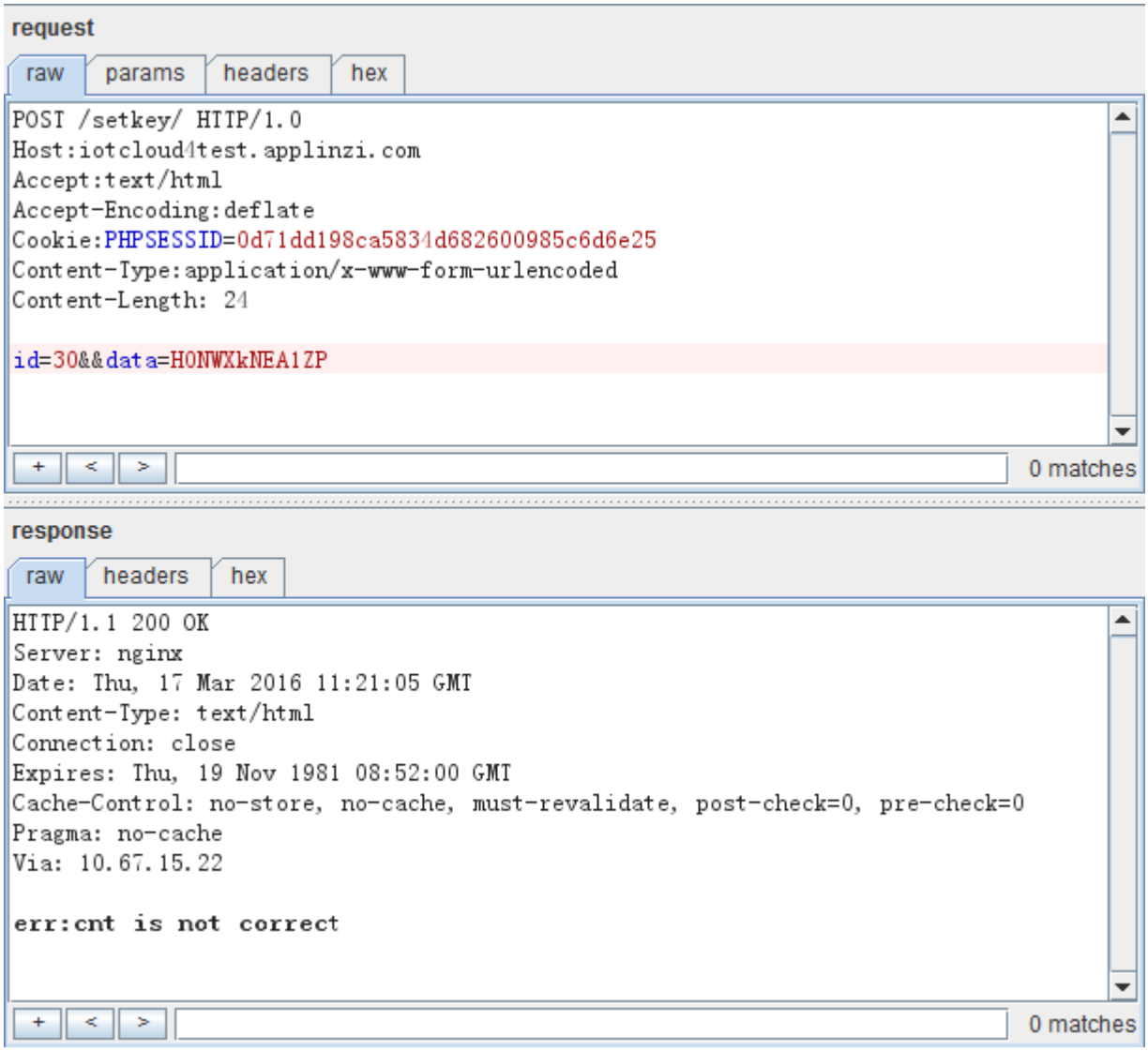

图 5.6 重放攻击及其响应

Fig. 5.6 Replay attacks and response

如图所示，服务器收到报文后，解码解密后发现计数器的值不大于之前上报的计数器值，故返回错误，并立即退出。

5.1.4 篡改攻击

攻击者将截获后的数据篡改再次发送到服务器。假设攻击者截获了物联网终端 ping 阶段的通信报文，并使用工具将截获后的报文篡改后发给服务器。图 5.7 展示了篡改攻击服务器的返回结果。

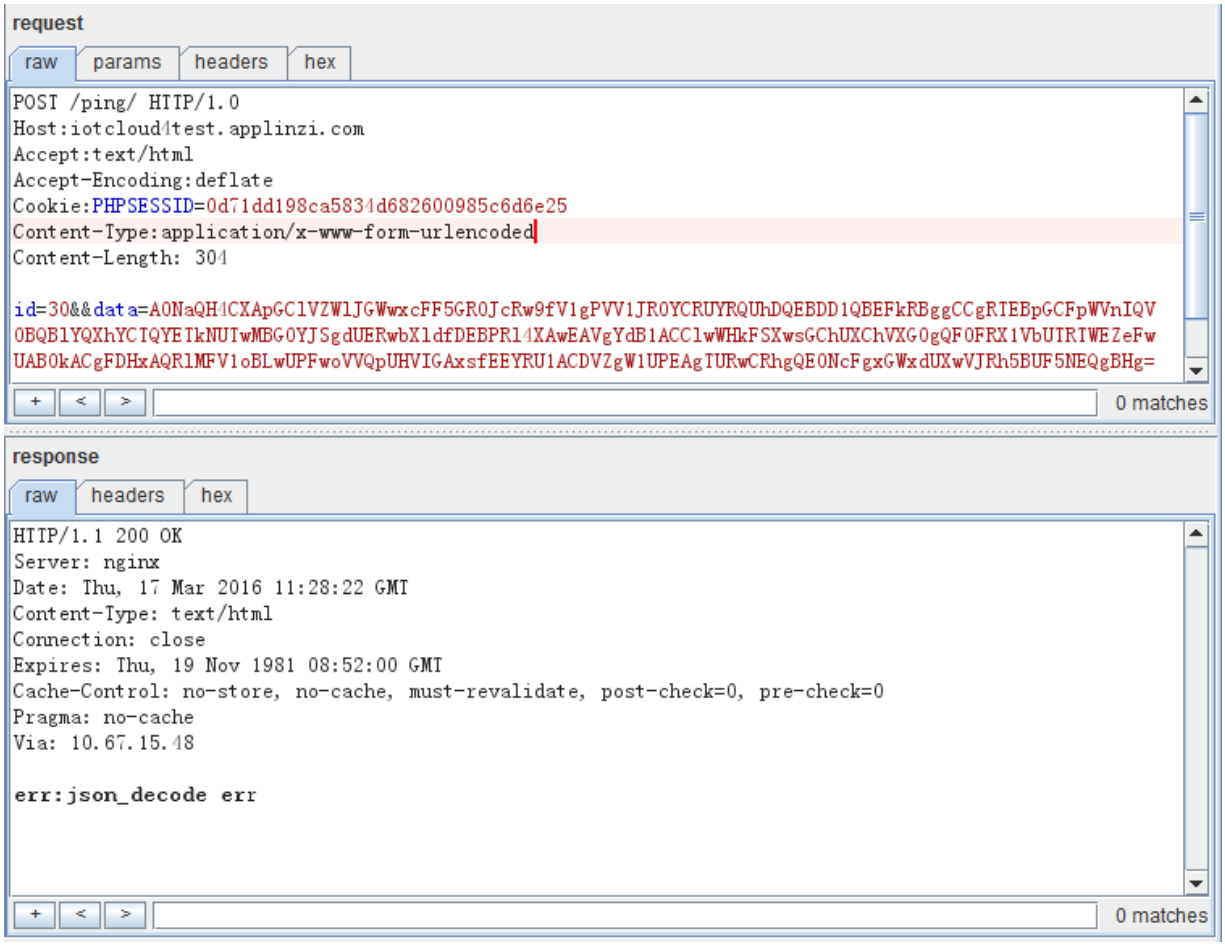

图 5.7 篡改攻击及其响应

Fig. 5.7 Tampering attacks and response

如图所示，由于 data 对应的数据已经被篡改，服务器将收到报文后解码解密后，发现不能使用 json_decode()函数解析，故返回错误，并立即退出。

5.1.5 伪造攻击

攻击者伪造一个假的物联网终端，与服务器通信。图 5.8 展示了攻击者伪造了一个不存在的物联网终端。

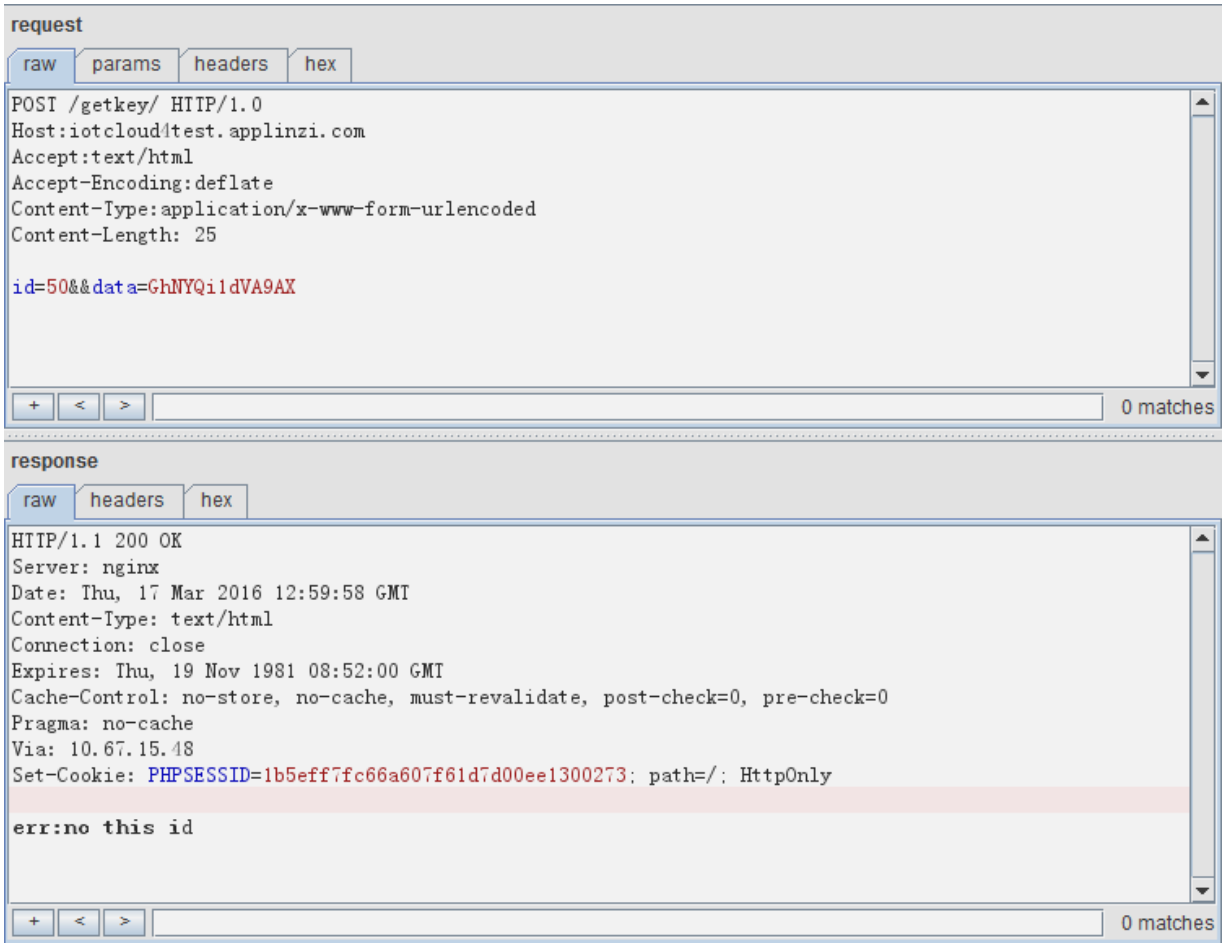

图 5.8 伪造攻击及其响应

Fig. 5.8 Forgery attacks and response

由于服务器中没有对应该设备 ID 的设备，故返回错误，并立即退出。

5.1.6 中断攻击

攻击者物联网终端破坏、或者将其移走后，后台会显示设备不在，并保存了设备最后一次心跳时间。如图 5.9 所示。

图 5.9 设备掉线

Fig. 5.9 Offline device

如图所示，管理系统显示了设备已掉线，只要在服务器中加入任务计划，每隔一段时间扫描一次设备，当检测到设备掉线时及时向相关人员发出警告信息，这样可以有效的找到设备掉线原因。这种掉线警告机制在一定程度上抵御了中断攻击。

5.3 本章小结

本章首先介绍了物联网终端程序的使用说明，之后按照使用方法，演示了物联网终端和服务器之间的正常通信流程。之后模拟了各种攻击，包括截获攻击、重放攻击、篡改攻击、伪造攻击、中断攻击等。通过对各种攻击的模拟从一定程度上验证了该模型的安全性能。

6 总结与展望

6.1 总结

本文首先分析了物联网终端接入网络常用的一种架构，“物联网终端-WEB 服务器”架构。该架构使用的是 HTTP 协议进行数据通信。通过对该架构的研究，为进一步研究基于 HTTP 的安全通信模型奠定了基础。

对于基于 HTTP 的安全通信模型的研究，本文首先举了市面上用到的几个实际案例，通过研究发现市面上针对安全通信使用的是 USERKEY 方式的安全通信模型，该模型明文传输的特点给攻击者带来了可操作空间。

针对 USERKEY 方式的安全通信模型的缺点，本文提出了一种新的模型，加密编码方式的安全通信模型，该模型的提出在不改变物联网终端与服务器通信架构的基础上，可以在一定程度上弥补 USERKEY 方式的安全通信模型的缺点。

在第四章中描述了如何结合实际应用场景对第三章中提出的理论模型加以实现，第五章演示了实现效果并模拟了多种攻击方式。通过模拟测试发现，新模型在一定程度上增加了物联网终端的安全性。

6.2 展望

物联网终端的安全包括各个方面，本文仅从物联网终端与服务器之间的通信环节上入手，提出一种新的理论模型。物联网终端安全还包括终端的设备自身的安全，例如攻击者使用特殊工具获取到了设备密码，那么它与服务器之间的通信将不再安全。另一方面，物联网管理系统的安全也是要考虑的，如果攻击者盗取了管理者的帐号和密码，将获取所有用户所管理的物联网终端的密码，在这种情况下，物联网终端与服务器之间的通信也将不再安全。总而言之，安全问题涉及到系统的每个环节，加强对每一个环节的安全工作才是保证系统安全的正确途径。

模型在克服了 USERKEY 方式的安全通信模型的缺点的的同时，也引入了一些缺点。例如通信流量的增加、设备运算量的增加、不利于物联网终端开发者调试等等。每个通信模型都具备两面性，各自具有其优缺点。只有抓住主要矛盾，合理的处理好其优缺点，才能在实际的应用场合中选择更合适的模型。

致 谢

三年的研究生生涯即将结束。三年间，我不仅巩固和提高了我的专业理论知识，也提高了学术研究能力以及动手实践能力。遇到了知识渊博的老师和很多专业知识精尖的同学，他们对我的人生产生了重要积极的影响。在此我向所有教导过我和帮助过我的人致以最崇高的敬意和真诚的感谢！

首先我想感谢的是我最尊敬的父母。他们虽没有太大的文化，却用他们最辛勤的劳动和最无私的爱，将我送到了知识的殿堂。用他们最真诚的付出和质朴善良的人格教会了做人的道理。不仅培养我良好的品格，还使我拥有了一颗感恩的心！在此祝愿我的父母身体健康，万事如意！

其次我要感谢我最敬爱的导师。他给我指明了学术研究的方向。在日常生活上，他们以身作则，教会了我很多为人处世的道理和做事的原则。在学术研究上，他知识渊博、治学严谨，使我在学习和科研上有了很大进步。在思想上，也能使我更加积极奋发向上。在此感谢我的导师对我的培养和关心，也祝愿老师的家人健康快乐！

最后要感谢所有教导过我的老师和帮助过我的同学，跟你们一起学习不仅使我学到了更广阔的知识，也使我收获了珍贵的友情。也要感谢在实习期间给我指导和帮助的所有的同事及领导，他们不仅教会了工作中的技能，也教会了我做人做事的道理和方式，使我更加容易的适应以后的工作。

攻读硕士学位期间主要研究成果

发表的论文：

嵌入式 Linux 的时钟语音芯片 YF017 驱动设计[J]. 单片机与嵌入式系统应用，2015, 15(10): 55-58.（第一作者）

参考文献

[1] 罗军舟,等.云计算:体系架构与关键技术[J].通信学报,2011,32(7):3-21.

[2] Dong Xiaoxia,Lu Tingjie.Review of the cloud computing and its future development[J].Journal of Beijing University of Posts and Telecommunications,2010,12(5):76-80.

[3] 钱志鸿,王义君.物联网技术与应用研究[J].电子学报,2012,40(5):1023-1029.

[4] 刘砚.基于云计算的物联网系统架构研究[J].科技信息,2012(1),209-210.

[5] 李志宇.物联网技术研究进展[J].计算机测量与控制,2012,20(6):1445-1448.

[6] 孙其博,等.物联网：概念、架构与关键技术研究综述[J].北京邮电大学学报,2010,33(3):1-9.

[7] 樊雪梅.物联网技术发展的研究与综述[J].计算机测量与控制.2011,19(5):1002-1004.

[8] 绳立成,等.我国物联网产业集群发展态势及路径分析[J].北京科技大学学报（社会科学版）,2010,26(3):138-141.

[9] http://wenku.baidu.com/view/58e6385f0722192e4436f66f.html[EB/OL].

[10] 汤明伟.浅谈 COOKIE 技术[J].常州信息职业技术学院学报,2005,4(1):46-48.

[11] 祝瑞,车敏.基于 HTTP 协议的服务器程序分析[J].现代电子技术,2012,35(4):117-122.

[12] 谢希仁.计算机网络(第 5 版)[M].北京:电子工业出版社,2011,243-244.

[13] 叶强.超文本传输协议:HTTP/1.0[J].科技情报开发与经济,2004(8):66-68.

[14] 肖戈林.HTTP 协议技术探析[J].江西通信科技,2001(1):39-41.

[15] 张宏烈,刘彦忠.基于套接字关于 HTTP 的研究[J].齐齐哈尔大学学报,2004(2):70-72.

[16] 李腊元,李春林.计算机网络技术[M].北京:国防工业出版社,2004.

[17] 王保云.物联网技术研究综述[J].电子测量与仪器学报,2009,23(12):1-7.

[18] 沈苏彬,等.物联网的体系结构与相关技术研究[J].南京邮电大学学报:自然科学版,2009,29(6):10-11.

[19] 侯琛,等.物联网中的嵌入式终端[J]. 电子测量技术,2014,37(10):113-118.

[20] 吴功宜.智慧的物联网:感知中国和世界的技术[M].北京:机械工业出版社,2010.

[21] 胡永利,等.物联网信息感知与交互技术[J].计算机学报,2012,35(6):1147-1163.

[22] 郭泽民.动态网页技术 PHP、ASP 与 JSP 的比较分析[J].煤炭技术,2007,26(6):29-30.

[23] 郭亮,高辉.动态网页技术 ASP,PHP,JSP 的比较[J].黑龙江科技信息,2007(17):80.

[24] 张铃丽,黄晓巧.动态网页开发技术研究[J].软件导刊,2010,9(1):126-128.

[25] 肖维明.基于 PHP+MySQL 的网站开发[J].物流工程与管理,2009,180(6):90-92.

[26] http://www.json.org[EB/OL].

[27] 郭武士.JSON 在 Web 开发中的应用[J].四川工程职业技术学院学报,2007,(3):41-43.

[28] 龚建华.JSON 格式数据在 Web 开发中的应用[J].办公自动化,2013,264(10):46-48.

[29] 朱继团.动态密码在保密信息系统中的应用[J].现代计算机(专业版),2004,189(6):41-44.

[30] 朱泽民.会话跟踪技术与应用[J].农业网络信息,2006,(7):57.

[31] 蒋长浩.php 专家指南[M].北京:中国电力出版社,2000.

[32] 郑江波.PHP 中 Session 机制的研究与应用[J],荆门职业技术学院学报,2007,22(3):37-40.

[33] 韩宇贞,朱华生.基于 Base64 编码的数据加密技术[J].南昌水专学报,2002,21(4):38-40.

[34] 唐武生,等.Base64 编码的实现与应用研究[J].长春大学学报,2006,16(2):69-72.

[35] 赵华峰.密码协议中重放攻击的研究[J].科学技术与工程,2008,8(18):5308-5311.

[36] 韩崇砚,等.一种抗重放攻击的 Web 服务认证协议[J].计算机工程,2011,37(21):91-93.

[37] 李俊灏.基于 OpenWRT 的多 WAN 口路由器[J].科技信息,2014,(5):83.

[38] https://openwrt.org/[EB/OL].

[39] 徐鹏,等.互联网应用 PaaS 平台体系结构[J].北京邮电大学学报,2012,35(1):120-124.

[40] Bu Wenjun,Zou Chaobin, Yu Jianlin.Hot cloud computing platform recommendations[J].Silicon Valley,2011,5:10-11.

[41] Cong Lei.Sina app engine architecture[J].Programmer,2010,11:59-62.

[42] 施伯乐.网页开发语言 JavaScript 实践教程[M].上海,上海交通大学出版社,2004,4.

[43] 张云苑.JavaScript 在动态网页设计中的应用[J].科技信息,2007,(5):23-24.

[44] 刘旭光.基于 AJAX 与 JAVASCRIPT 技术在网页中传递数据的实现[J].淮北煤炭师范学院学报(自然科学版),2010,31(1):51-55.

[45] 屈展,李婵.JSON 在 Ajax 数据交换中的应用研究[J].西安石油大学学报(自然科学版),2011,26(1):95-98.

[46] 熊文,等.Ajax 技术在 Web2.0 网站设计中的应用研究[J].计算机技术与发展,2012,22(3):145-148.

[47] 游丽贞,等.Ajax 引擎的原理和应用[J].微计算机信息,2006,22(2):205-207.