2011 第三届 OpenHW 开源硬件大赛
项目论文

项目名称：面向无线传感器的低功耗密码芯片设计

学校名称：华南理工大学

指导老师：唐韶华　　赖晓铮

网址：http://www.openhw.org

视频展示链接：

http://v.youku.com/v_show/id_XMzc0NDkyMzY4.html

目录

1.前言

近年来，物联网的提出带动了一股物联网发展的浪潮。2009 年 IBM 提出了基于物联网的智慧地球的概念，而美国则把智慧地球当作是经济振兴计划的一个核心环节。美国 Forester 预测，到 2020 年，世界上物互联的业务，跟人与人通信的业务相比，将达到 30:1，物联网将成为下一个万亿级的通信业务。在国内，2009 年，温家宝总理明确指出“要着力突破传感网、物联网关键技术，及早部署后 IP 时代相关技术研发，使信息网络产业成为推动产业升级、迈向信息社会的‘发动机’”。2010 年，物联网等新兴战略产业首次被写入国务院政府工作报告。到 2020 年前，中国规划了 3.86 万亿元的资金用于物联网产业化的发展。这说明物联网的发展前景不管是在国内国外，都具有巨大的潜力。

而物联网的实现要立足于信息的采集以及处理，信息采集主要通过 RFID（射频识别）电子标签和传感器实现，其中，物联网通过无线传感器网络（WSN）来实现真实世界中物体的调度和部署。无线传感器网络(WSN) 是一种大规模的自组织网络，通过大量低成本、资源极度受限的传感节点设备协同工作，实现某一特定任务。由于 WSN 通常部署在无人维护、不可控制的环境中，使 WSN 除了具有一般无线网络所面临的信息泄露、信息篡改、重放攻击、拒绝服务等多种威胁外，而其中最重要的是信息的真实性，**如何保证传感网络以及所采集数据的真实性成为一个迫切需要解决的问题。**为了保证传感网络的安全，需要设计合适的密码芯片。**而密码芯片对于密码算法的资源受限较大，如何既保证密码算法的安全性，同时又保证其能适用于低功耗的芯片中，也成为一个难题。**

传统的公钥密码体系，以 RSA 为典范，它很好地解决了传统对称密码体系里面遇到的密钥交换问题，但是，它的缺点也逐渐暴露。首先，它的安全性是建立在大整数因数分解的复杂性上的，而根据近年来在整数因数分解上的研究，为了保证其安全性，不得不使用较大的参数，从而降低了加解密计算效率。另外，量子计算机的出现，直接威胁到了 RSA 算法的安全。因此，急需寻找一种新的安全高效的密码体系来代替现有的密码体系。

MPKC(Multivariate Public Key Cryptosystems，多变量公钥密码系统)，是基于在有限域上多变量函数的密码体制，是一种新的研究方向，用以代替类似 RSA 这种基于数论的公钥密码系统。它的安全性是基于有限域上多变量方程组的求解问题，即 MQ 问题。有限域上多变量方程组的求解问题在 1979 年被 Garey 和 Johnson 证明是 NP(非多项式)难度的问题，基于 MQ 的公钥密码体制首先在 1988 年由 Matsumoto 和 Imai 提出，而量子计算机本身在处理 NP 难度的问题上并没体现出优势。相反，大整数因式分解并没有被证明是 NP 难度的问题，确切地说，它没有被证明是何种等级难度的问题。另外，量子计算机可以很好地解决大整数因式分解的问题。而本文所设计的 rainbow 算法就是 MPKC 的一种标准。

本项目通过空气质量监测、红外温度传感器、Nexys3 FPGA 开发、无线通讯技术、云端平台和社交网络的有机融合，构成了一个高度可信的无线监控信息实时发布平台。该平台集火灾监测、空气质量监测、社交网络互动、云端信息发布、短信报警、电子邮件提醒等功能于一体。能够实时监控当地的温度、空气质量，并签名后传输数据，签名算法采用新型的公钥密码算法 Rainbow 实现低功耗的密码芯片来保证数据的真实性和可靠性，从而构成了一个可信度极高的数据采集和发布平台，实现了基于物联网的可信赖的、安全的云平台。

2.总体方案设计

2.1 整体架构

如图 2-1 所示，整个系统方案分为三层。在中间层，我们通过温度传感器和空气质量监控模块采集数据，系统控制模块负责将采集的数据送往底层 IP 核。底层 IP 核通过新型公钥密码算法 Rainbow 进行数据签名，并将签名数据返回上层给系统控制模块。然后系统控制模块 GPRS/GSM 模块进行数据传输到上层。上层通过云平台可以实现超远距离的监控，利用新浪 SAE 云端实现数据的认证、Web 端发布数据以及利用新浪微博进行社区互动。

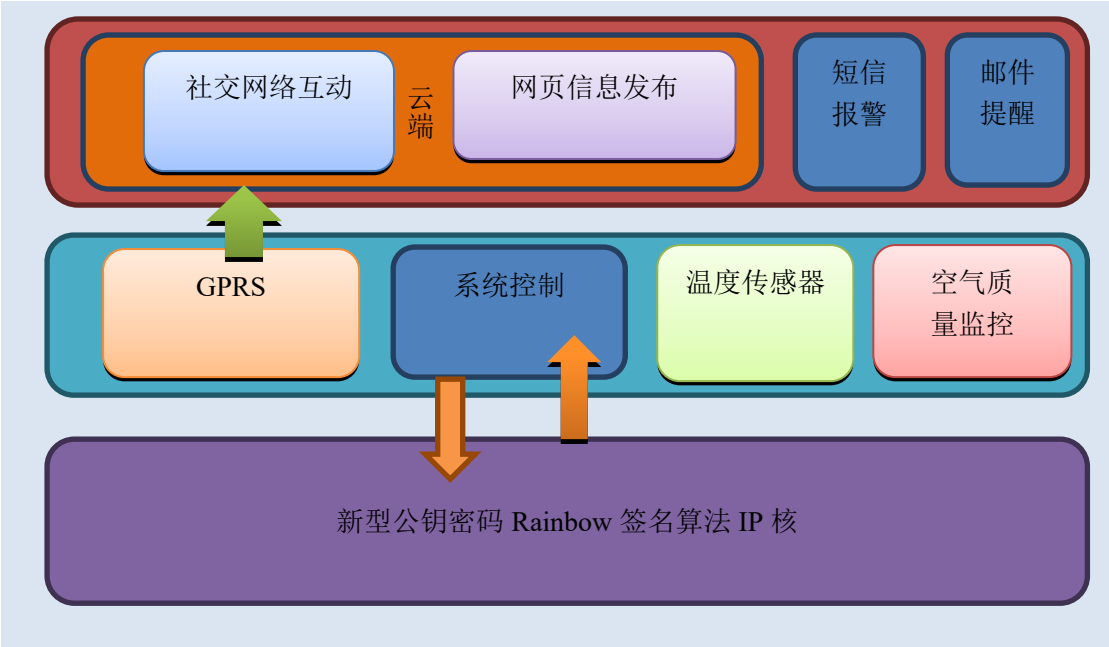


图 2-1 系统整体架构

2.2 密码方案

2.2.1 多变量公钥密码的优势

为了保证传感网络的安全，需要设计合适的密码芯片。而密码芯片对于密码算法的资源受限较大，如何既保证密码算法的安全性，同时又保证其能适用于低功耗的芯片中，也成为一个难题。

传统的公钥密码体系，以 RSA 为典范，它很好地解决了传统对称密码体系里面遇到的密钥交换问题，但是，它的缺点也逐渐暴露。首先，它的安全性是建立在大整数因数分解的复杂性上的，而根据近年来在整数因数分解上的研究，为了保证其安全性，不得不使用较大的参数，从而降低了加解密计算效率。另外，量子计算机的出现，直接威胁到了 RSA 算法的安全。因此，急需寻找一种新的安全高效的密码体系来代替现有的密码体系。

MPKC(Multivariate Public Key Cryptosystems，多变量公钥密码系统)，是基于在有限域上多变量函数的密码体制，是一种新的研究方向，用以代替类似 RSA 这种基于数论的公钥密码系统。它的安全性是基于有限域上多变量方程组的求解问题，即 MQ 问

题。有限域上多变量方程组的求解问题在 1979 年被 Garey 和 Johnson 证明是 NP(非多项式)难度的问题，基于 MQ 的公钥密码体制首先在 1988 年由 Matsumoto 和 Imai 提出，而量子计算机本身在处理 NP 难度的问题上并没体现出优势。相反，大整数因式分解并没有被证明是 NP 难度的问题，确切地说，它没有被证明是何种等级难度的问题。另外，量子计算机可以很好地解决大整数因式分解的问题。而本文所设计的 rainbow 算法就是 MPKC 的一种标准。

因此，我们采用 Rainbow 算法来实现面向无线传感器的低功耗密码芯片设计。

2.2.2 MPKC 密码体系简介

目前，有几种方案可以作为 RSA 的替代方向。主要有，点的集合建立在阿贝尔群结构上的椭圆曲线密码学、以格为度量的基于格的密码学、基于纠错编码的密码学。另外一种就是 MPKC（multivariate public key cryptography，多变量公钥密码学），它是建立在有限域上的多项式计算基础上的，并且这些多项式的度均为二，即平方多项式。由于在有限域中，求解一组多变量多项式方程组被证明是一个 NP-hard 的问题，并且量子计算机在解决 NP-hard 的问题方面并没有体现出什么优点，因此 MPKC 得到了巨大的支持和推进。

2.2 .3 MPKC 签名标准

目前，关于 MPKC 的构造标准已有很多种。如油醋签名标准、三角标准等。后面将会介绍 Rainbow 签名标准，它属于油醋签名标准的一种拓展标准，本次比赛项目就是采用 Rainbow 的签名标准。

2.2.4 基本的油醋签名策略

油醋签名标准[1]的基本结构就是油醋多项式。即在多项式里面分有油变量和醋变量，油醋多项式是油变量为线性的（即最高次幂为 1）平方多项式。在选取好所有醋变量的值后，平方油醋多项式就变成了变量为油变量的线性多项式。通过一组油醋多项式，就可以产生一个签名。

设 k 为一个有 q 个元素的有限域。变量 $x_1,\ldots,x_o$ 称为油变量，变量 $\overset{v}{x}_1,\ldots,\overset{v}{x}_v$ 称为醋变量。并且 n = o + v。

一个油醋多项式的定义为，任何满足下列形式且度为 2 的多项式：

$$f = \sum_{i=1}^{o}\sum_{j=1}^{v} a_{ij} x_i \overset{v}{x}_j + \sum_{i=1}^{v}\sum_{j=1}^{v} b_{ij} \overset{v}{x}_i \overset{v}{x}_j + \sum_{i=1}^{o} c_i x_i + \sum_{j=1}^{v} d_j \overset{v}{x}_j + e$$

其中，a_{ij}，b_{ij}，c_i，d_i，e 在有限域 k 上，$f \in \mathrm{k}[x_1,\ldots,x_o,\overset{v}{x}_1,\ldots,\overset{v}{x}_v]$。

按照以上对油醋多项式的定义，选取 o 个油醋多项式，$f_1,\ldots,f_o$。签名时，先将要签名的消息摘要 M 表示成 M($\alpha_1,\ldots,\alpha_o$)。再随机选取醋变量值($\overset{v}{x}_1',\ldots,\overset{v}{x}_v'$)。代入求解线性方程组：

$f_1(x_1,...,x_o,x_1^{v'},...,x_v^{v'})=\alpha_1$

$f_2(x_1,...,x_o,x_1^{v'},...,x_v^{v'})=\alpha_2$

.

.

.

$f_O(x_1,...,x_o,x_1^{v'},...,x_v^{v'})=\alpha_O$

得到油变量$x_1,...,x_o$的值。注意，这里解方程可能无解，但是通过更改随机选取的$x_1^{v'},...,x_v^{v'}$可以很快找到有解的方程组。因此，可以获得$\vec{X}=(x_1,...,x_o,x_1^v,...,x_v^v)$的值。然后再选取一个可逆矩阵 L。使得，$\vec{X}$=L∘Z，即 Z=$L^{-1}\circ\vec{X}=(Z_1,...,Z_n)^T$。其中，n=o+v。$(Z_1,...,Z_n)^T$就是签名后的消息。认证时，令 F($Z_1,...,Z_n$)=($\overline{f}_1,...,\overline{f}_o$)，即 F 代表 o 个变量为$Z_1,...,Z_n$的多项式，其中多项式$\overline{f}_i=f_i\circ$ L。将签名后的消息($Z_1,...,Z_n$)代入 F，可以计算得到($\alpha_1',...,\alpha_o'$)，再比较其与消息摘要 M($\alpha_1,...,\alpha_o$)是否相同，就可以验证签名是否正确。

在这里可以看出，私钥为$f_1,...,f_o$和 L，以及有限域 k 的构造。公钥就是$\overline{f}_1,...,\overline{f}_o$。

2.2.5 Rainbow 的基本原理

下面介绍 Rainbow 签名的基本原理。Rainbow 其实是基本的油醋签名标准的一种拓展。在油醋签名标准中，油变量和醋变量的个数 o 和 v 相等时，称为平衡油醋，不相等时，称为非平衡油醋。经过对油醋签名标准的安全研究发现，要保证签名算法的安全性，必须采用一种多层的非平衡油醋结构来构造油醋签名，这种签名标准被称为 Rainbow 签名标准。

接着来介绍下 Rainbow 的总体结构。

令 S 为集合{1,2 ,3 ,...,n}。令$v_1,...,v_u$为关于整数 u 的集合，其中 u≤n，并且 0<$v_1<v_2<$ ...<v_u=n。再定义整数集合S_l={1,2 ,...,v_l}，l=1,...,u。从而有$S_1\subset S_2\subset\cdots\subset S_u=S$，$v_i$为$S_i$中元素的个数。

令$o_i=v_{i+1}-v_i$且$O_i=S_{i+1}-S_i$，i=1,...,u-1。因此，o_i是集合O_i的元素个数。根据这

个概念，定义线性平方多项式 P_l 为中心映射：

$$\sum_{i\in O_l,\, j\in S_l} \alpha_{ij} x_i x_j + \sum_{i,j\in S_l} \beta_{ij} x_i x_j + \sum_{i\in S_{l+1}} \gamma_i x_i + \eta$$

可以看出，这就是前面定义的油醋多项式类型，其中 x_i 是油变量，x_j 是醋变量。并且，称 x_i 是一个第 l 层的油变量，当 $i \in O_l$ 时；称 x_j 是一个第 l 层的醋变量，当 $j \in S_l$ 时。一个多项式 P_l 称为第 l 层油醋多项式。显然，当 $i < j$ 时，$\mathrm{P_i} \subset \mathrm{P_j}$，且多项式集合 $\{P_1, \ldots, P_{u-1}\}$ 是一个油醋多项式集合。而第 l +1 层的醋变量为第 l 层的油醋变量，因为 $S_{i+1} = O_i \cup S_i$。

根据以上定义，再定义映射 F 为 n-v_1 个多项式 $F_1, \ldots, F_{n-v_1}$。F_i 是随机从 P_i 中选取的油醋多项式。因此 F 是 u-1 层的油醋多项式结构。第一层包含 o_1 个多项式 $F_1, \ldots, F_{o_1}$，其中，$\{x_i \mid i \in O_1\}$ 是油变量，$\{x_j \mid j \in S_1\}$ 是醋变量。第 l 层包含 o_l 个多项式 $F_{v_l+1}, \ldots, F_{v_{l+1}}$，其中，$\{x_i \mid i \in O_l\}$ 是油变量，$\{x_j \mid j \in S_l\}$ 是醋变量。

每层的变量如图 2-7 所示，先选 v_1 和 o_1，再根据公式 $v_{i+1} = v_i + o_i$ 计算 v_2，然后再选取 o_2，依此类推，确定总共 u-1 层的 v 和 o 的大小。称这 u-1 层的变量为彩虹（Rainbow）变量：

$$[x_1, \ldots, x_{v_1}], \{x_{v_1+1}, \ldots, x_{v_2}\};$$
$$[x_1, \ldots, x_{v_1}, x_{v_1+1}, \ldots, x_{v_2}], \{x_{v_2+1}, \ldots, x_{v_3}\};$$
$$[x_1, \ldots, x_{v_1}, x_{v_1+1}, \ldots, x_{v_2}, x_{v_2+1}, \ldots, x_{v_3}], \{x_{v_3+1}, \ldots, x_{v_4}\};$$
$$\vdots$$
$$[x_1, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, \ldots, x_{v_{u-1}}], \{x_{v_{u-1}+1}, \ldots, x_n\}$$

接着，再选取两个随机可逆映射 L_1：$k^{n-v_1} \to k^{n-v_1}$ 和 L_2：$k^n \to k^n$，然后定义 $\overline{F} = L_1 \circ \mathrm{F} \circ L_2$。

最后，可以得到公钥为有限域 k 的构造方式和多项式 $\overline{F}$。私钥为 F，L_1 和 L_2。

3.系统硬件设计

3.1 硬件架构

无线终端由红外温度传感器、空气质量监控模块、Nexys 3 FPGA 开发板、GPRS/GSM 模块组成。FPGA 开发板上实现以软核 Microblaze 为 MCU 的嵌入式控制系统和新型公钥密码算法 Rainbow 芯片。经红外温度传感器和空气质量模块采集数据后，使用实现的密码算法芯片进行签名后，再通过 GPRS/GSM 模块以 HTTP 请求的方式发送的云端数据处理平台。发送到云端处理平台的数据是签名后的位置信息（固定）、温度信息以及空气质量信息。

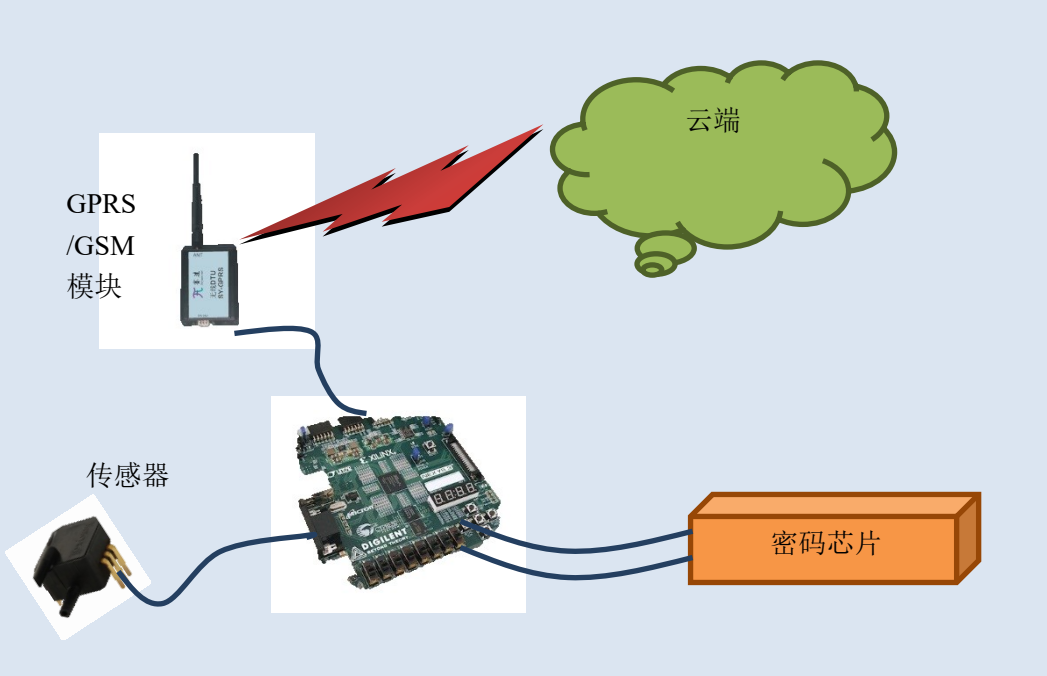


图 3-1 系统硬件架构

3.2 无线终端的硬件实现

无线终端的主要功能就是通过传感器采集数据，签名后通过 GPRS/GSM 模块把数据发送到云端服务器上，并且发送带有数据的提示 Email。报警时，额外发送报警短信到相应部分的短信报警平台。硬件方面，主要使用了 PTM 100 GPRS/GSM 模块、空气质量模块、MLX90614 红外温度传感器。软件方面，主要用到了 UART、IIC、GPIO 时钟以及自设计的 Rainbow 密码芯片等资源。软件的具体实现请参见 4.2 节。

无线终端与云端的通讯利用 GPRS/GSM 模块实现，使用了 GPRS 模块的短信功能以及 TCP 连接功能。Nexys3 内部的控制系统中，MCU 为 Microblaze、总线为 AXI-lite、内存有片上资源生成的内存模块以及 Nexys 板载的 16MB Cellular RAM 构成，中断控制器直接使用 Xilinx 提供的 IP 核生成。而外设方面，使用了 3 个 Uart-Lite IP 核，分别作为系统标准输入输出、GPRS 控制器、蓝牙控制器（暂未使用）；1 个 2 位的 gpio 控制器用来读取空气质量传感模块的数据；1 个 IIC 总线控制器来读取红外温度传感器的数据。如图 3-2 所示：

GPRS 模块与 Nexy3 的通讯使用 UART-LITE IP 核，其中 UART-LITE IP 核中系统布线

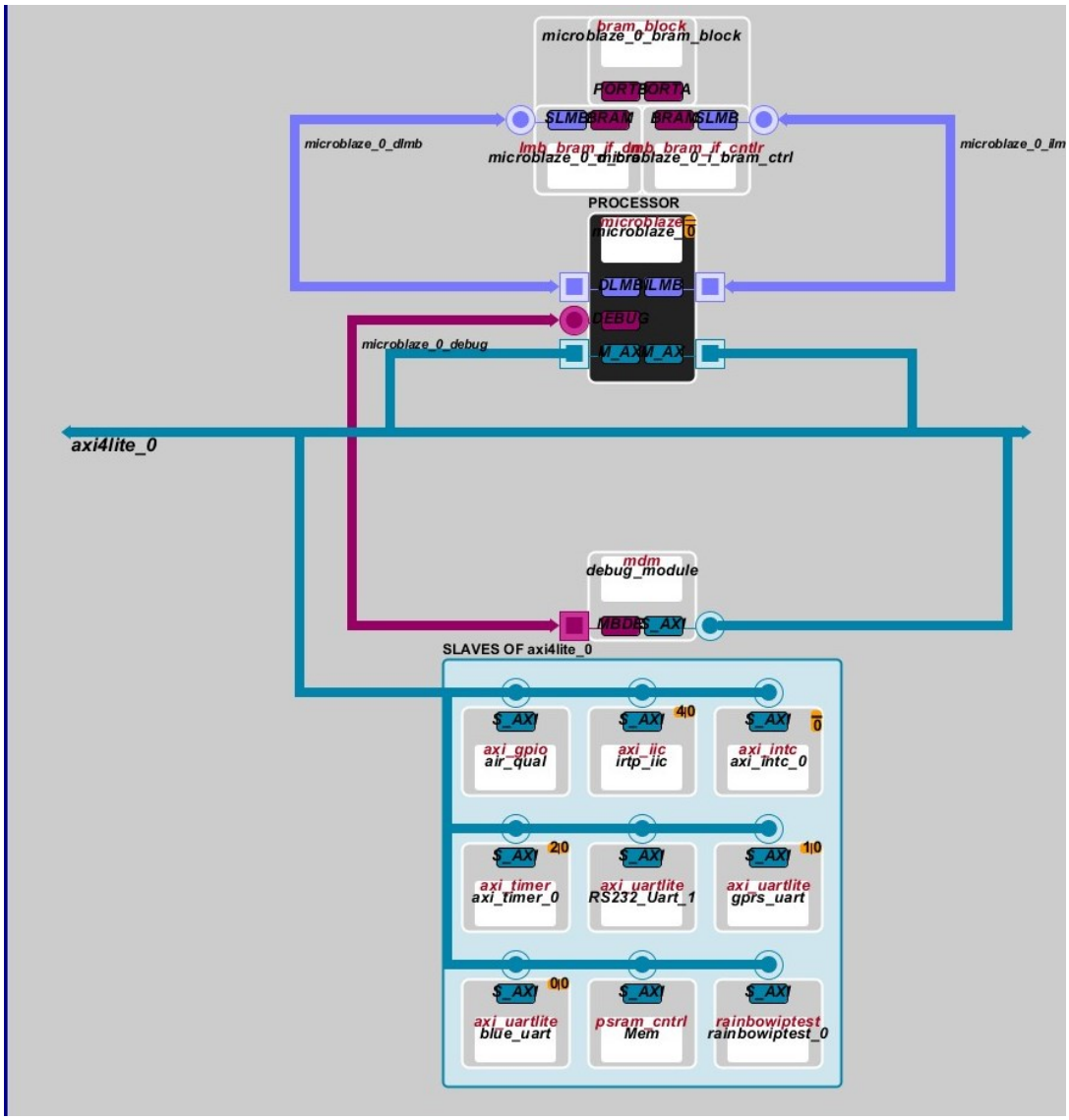


图 3-2 系统架构图

的两根管脚 RX 接 GPRS 模块的 TX，TX 接 GPRS 模块的 RX，由于 GPRS 模块使用的电源是 5v 的 TTL 电平，因此我们电源采用板上 J11 的两根管脚（与外部供电电源接口并联，考虑到所用外设较多，因此我们对 Nexys3 采用 5V 的外部电源供电）。

无线终端的温度数据采集是通过 MLX-90614ESF-BCC 非接触式红外温度传感器，MLX-90614ESF-BCC 连接如图 3-3 所示。该传感器与 Nexys3 的通讯采用 IIC 总线协议，其中 AXI-IIC IP 中指定的 SDA、SCL 分别接红外传感器的 SDA 和 SCL（红外传感器本身是兼容 SMBUS 总线协议，但考虑到 SMBUS 与频率在 10KHz 到 100KHZ 之间的 IIC 总线协议是兼容的，限于 Xilinx 并没有提供 SMBUS 的 IP 核，因此我们采用 IIC IP 核与温度传感器进行通讯），供电采用板上 Pmod 接口提供的 CMOS 3v3 的电平。空气质量数据的采集是通过深圳是戴维莱公司的 TPM-401P 空气质量传感模块，与 Nexys3 的通讯采用普通 IO 口进行通讯，其中传感模块的 AB 接口分别接 Nexys3 上 Xilinx 提供的 GPIO IP 定义的两根管脚。供电与 GPRS 模块一样，采用 5v 的 TTL 电平。GPRS 模块、空气质量模块连接如图 3-4 所示。

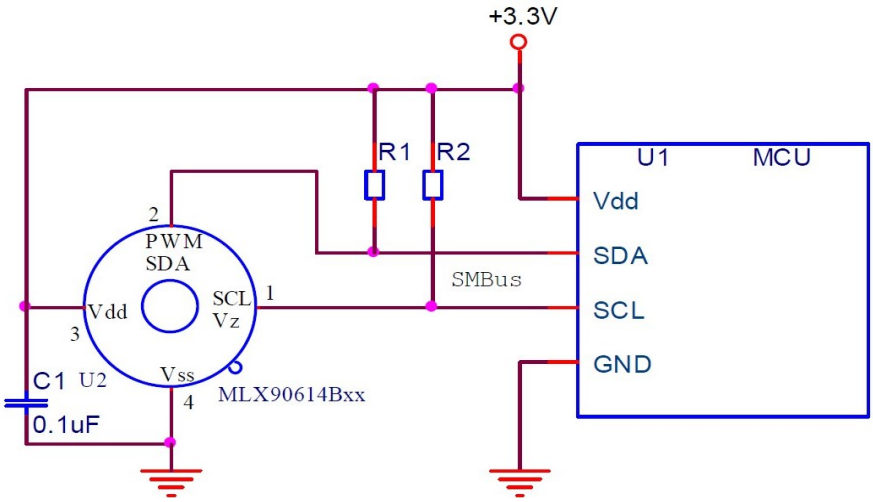


图 3-3MLX-90614ESF-BCC 连接示意图

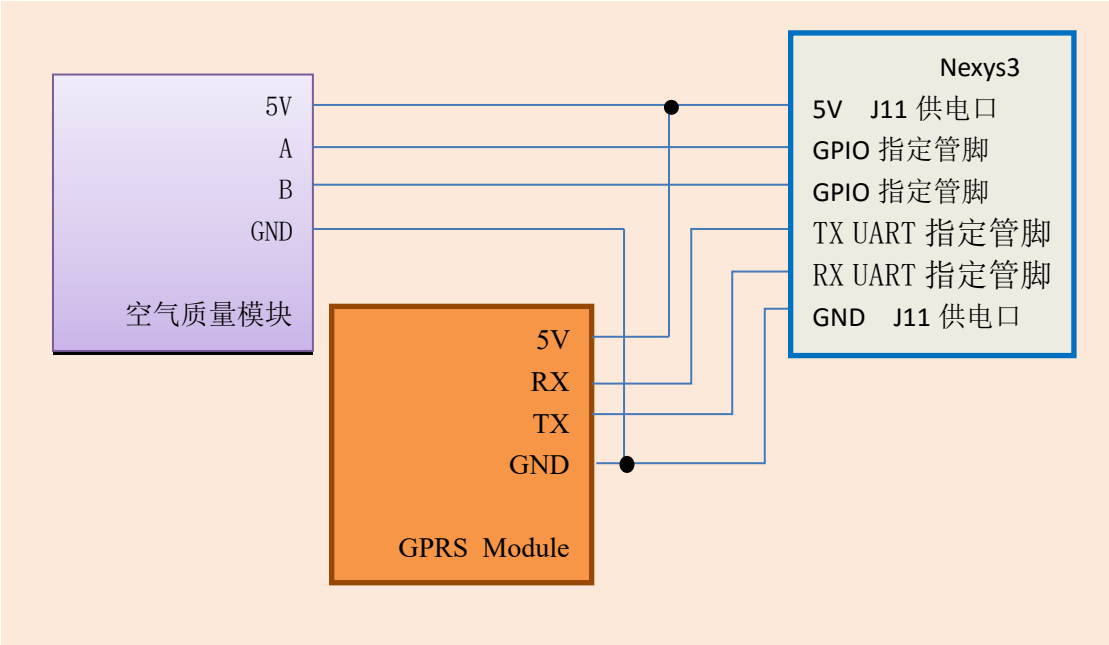


图 3-4 GPRS 模块、空气质量模块连接示意图

3.3 Rainbow 签名算法的 FPGA 实现

3.3.1 签名算法的状态机实现

Rainbow 签名算法的实现采用状态机的方式并将私钥存放于 rom 里面，一共分为了 44 个状态。其中，初始状态为 idle，当有数据输入并将外部触发信号 start 置 1 时，读入输入的数据并将状态从 idle 转为 init，开始签名算法。接着开始读入 rom 里面的私钥和计算签名的过程，这个过程在 40 个中间状态间进行切换运算。直到签名结束，状态转换为 signFinish，此时输出签名结束信号和签名结果并进入 signStop，等待外围电路读取数据完毕，将信号 start 置 0，进入 idle 状态。状态机如下图所示：

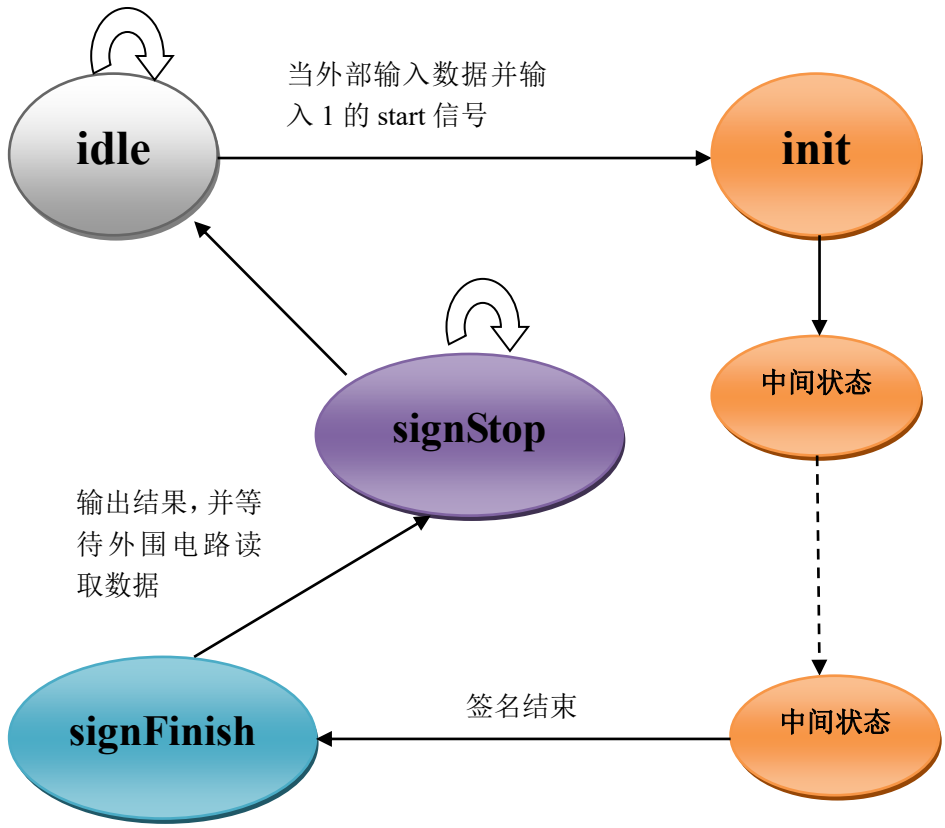


图 3-5 状态机实现 rainbow 过程

3.3.2 有限域的性质

Rainbow 签名算法涉及的所有基本运算都是在有限域 GF(2^8)上得。有限域上元素的表示方式有 4 种。第一种是二进制表示，将 256 个域上的元素表示成 8bit 的二进制数。第二种方式是表示成十进制 0 到 255。第三种方式是采用多项式表示。例如，一个 8 位的二进制数 10101010 能表示成多项式$x^7+x^5+x^3+x$，多项式的最高次项的幂小于 8 并且系数为 0 或 1。最后一种是采用指数表达式的形式。假设 f(x)是一个既约多项式，a 是一个基本表达式。令 x=a，则 $a^2=x^2,\ldots,a^7=x^7,a^8=x^8 \bmod f(x),\ldots,a^{255}=x^{255} \bmod f(x)$。因此，我们可以采用 a 的

不同指数表达式来表示有限域上的不同元素。

Rainbow 签名算法涉及到的核心基本运算器件设计有 GF(2^8)域上乘法器设计、乘法求逆器设计以及高斯消元算法。这里，我们采用有限域上元素的第四种表示方法。

3.3.3 GF(2^8)域上乘法器设计

GF(2^8)域上两个元素 A、B 及其乘积 C 可分别用多项式表示为：

$A(Z) = A_7Z^7+A_6Z^6+A_5Z^5+A_4Z^4+A_3Z^3+A_2Z^2+A_1Z+A_0$

$B(Z) = B_7Z^7+B_6Z^6+B_5Z^5+B_4Z^4+B_3Z^3+B_2Z^2+B_1Z+B_0$

$C(Z) = C_7Z^7+C_6Z^6+C_5Z^5+C_4Z^4+C_3Z^3+C_2Z^2+C_1Z+C_0$

其中，系数A_i、B_i、C_i ∈GF(2),i=0,1,...,7。A 与 B 的乘积可以分两步得到：首先 A(Z)和 B(Z)两个多项式按常规方法相乘，得到一个次数不大于 14 的多项式 M(Z)，然后将 M(Z)对既约多项式 P(Z)=$Z^8+Z^7+Z^2+Z+1$求模，得到次数不大于 7 的多项式，即 A 与 B 的乘积 C。根据这个原理，可用 verilog 描述乘法器的行为，下图是该乘法器的 verilog 功能描述。

```
module GF_mult(a,b,c
    );
input [7:0] a;
input [7:0] b;
output [7:0] c;
wire[7:0] c;
wire[14:0] ly;
wire[7:0] lym;
assign ly[14] = a[7] & b[7];
assign ly[13] = (a[7] & b[6]) ^ (a[6] & b[7]);
assign ly[12] = (a[7] & b[5]) ^ (a[6] & b[6]) ^ (a[5] & b[7]);
assign ly[11] =(a[7] & b[4])^(a[6] & b[5])^(a[5] & b[6])^(a[4] & b[7]);
assign ly[10] =(a[7] & b[3])^(a[6] & b[4])^(a[5] & b[5])^(a[4] & b[6])^(a[3] & b[7]);
assign ly[9] =(a[7] & b[2])^(a[6] & b[3])^(a[5] & b[4])^(a[4] & b[5])^(a[3] & b[6])^(a[2] & b[7]);
assign ly[8] =(a[7] & b[1])^(a[6] & b[2])^(a[5] & b[3])^(a[4] & b[4])^(a[3] & b[5])^(a[2] & b[6])^(a[1] & b[7]);
assign ly[7] =(a[7] & b[0])^(a[6] & b[1])^(a[5] & b[2])^(a[4]& b[3])^(a[3] & b[4])^(a[2] & b[5])^(a[1] & b[6])^(a[0] & b[7]);
assign ly[6] =(a[6] & b[0])^(a[5] & b[1])^(a[4] & b[2])^(a[3] & b[3])^(a[2] & b[4])^(a[1] & b[5])^(a[0] & b[6]);
assign ly[5] =(a[5] & b[0])^(a[4] & b[1])^(a[3] & b[2])^(a[2] & b[3])^(a[1] & b[4])^(a[0] & b[5]);
assign ly[4] =(a[4] & b[0])^(a[3] & b[1])^(a[2] & b[2])^(a[1] & b[3])^(a[0] & b[4]);
assign ly[3] =(a[3] & b[0])^(a[2] & b[1])^(a[1] & b[2])^(a[0] & b[3]);
assign ly[2] =(a[2] & b[0])^(a[1] & b[1])^(a[0] & b[2]);
assign ly[1] =(a[1] & b[0])^(a[0] & b[1]);
assign ly[0] =(a[0] & b[0]);
assign lym[7] = ly[7] ^ ly[9] ^ ly[11] ^ ly[12];
assign lym[6] = ly[6] ^ ly[8] ^ ly[10] ^ ly[11];
assign lym[5] = ly[5] ^ ly[10] ^ ly[11] ^ ly[12] ^ ly[14];
assign lym[4] = ly[4] ^ ly[9] ^ ly[10] ^ ly[11] ^ ly[13] ^ ly[14];
assign lym[3] = ly[3] ^ ly[8] ^ ly[9] ^ ly[10] ^ ly[12] ^ ly[13] ^ ly[14];
assign lym[2] = ly[2] ^ ly[8] ^ ly[13] ^ ly[14];
assign lym[1] = ly[1] ^ ly[9] ^ ly[11] ^ ly[13] ^ ly[14];
assign lym[0] = ly[0] ^ ly[8] ^ ly[10] ^ ly[12] ^ ly[13];
assign c = lym;

endmodule
```

3.3.4 GF(2^8)域上求逆器的设计

根据有限域性质，GF(2^8)上任一元素 a，有 a=a^{2^8}，因此$a^{-1}=a^{2^8-2}$，而$2^8-2=2+2^2+2^3+2^4+2^5+2^6+2^7$，所以$a^{-1}=a^{2+2^2+2^3+2^4+2^5+2^6+2^7}$，即

$a^{-1} = a^2 a^{2^2} a^{2^3} a^{2^4} a^{2^5} a^{2^6} a^{2^7}$，因此求逆器可用转换为相应的乘法，其硬件实现如下图：

```
module GF_inverse(inB,BInvert
    );
input [7:0] inB;
output [7:0] BInvert;
wire [7:0] B2;
wire [7:0] B4;
wire [7:0] B8;
wire [7:0] B16;
wire [7:0] B32;
wire [7:0] B64;
wire [7:0] B128;
wire [7:0] B2B4;
wire [7:0] B8B16;
wire [7:0] B32B64;
wire [7:0] B2B4B8B16;
wire [7:0] B32B64B128;

assign B2[0]= inB[0] ^ inB[4] ^ inB[5] ^ inB[6];
assign B2[1]= inB[7];
assign B2[2]= inB[1] ^ inB[4] ^ inB[7];
assign B2[3]= inB[4] ^ inB[5] ^ inB[6] ^ inB[7];
assign B2[4]= inB[2] ^ inB[5] ^ inB[7];
assign B2[5]= inB[5] ^ inB[6] ^ inB[7];
assign B2[6]= inB[3] ^ inB[4] ^ inB[5];
assign B2[7]= inB[6];
assign B4[0]= inB[0] ^ inB[2] ^ inB[3];
assign B4[1]= inB[6];
assign B4[2]= inB[2] ^ inB[5] ^ inB[6];
assign B4[3]= inB[2] ^ inB[3] ^ inB[4] ^ inB[5];
assign B4[4]= inB[1] ^ inB[4] ^ inB[5];
assign B4[5]= inB[3] ^ inB[4] ^ inB[7];
assign B4[6]= inB[2] ^ inB[4] ^ inB[5] ^ inB[7];
assign B4[7]= inB[3] ^ inB[4] ^ inB[5];
assign B8[0]= inB[0] ^ inB[1] ^ inB[4];
assign B8[1]= inB[3] ^ inB[4] ^ inB[5];
assign B8[2]= inB[1] ^ inB[3] ^ inB[6];
assign B8[3]= inB[1] ^ inB[2] ^ inB[5];
```

```
assign B8[4]= inB[2] ^ inB[6] ^ inB[7];
assign B8[5]= inB[2] ^ inB[4];
assign B8[6]= inB[1] ^ inB[2] ^ inB[4] ^ inB[7];
assign B8[7]= inB[2] ^ inB[4] ^ inB[5] ^ inB[7];
assign B16[0]= inB[0] ^ inB[2] ^ inB[4] ^ inB[6];
assign B16[1]= inB[2] ^ inB[4] ^ inB[5] ^ inB[7];
assign B16[2]= inB[3] ^ inB[6];
assign B16[3]= inB[1] ^ inB[4] ^ inB[5] ^ inB[6] ^ inB[7];
assign B16[4]= inB[1] ^ inB[3] ^ inB[5] ^ inB[6] ^ inB[7];
assign B16[5]= inB[1] ^ inB[2] ^ inB[4] ^ inB[5];
assign B16[6]= inB[1] ^ inB[2] ^ inB[4] ^ inB[5] ^ inB[6] ^ inB[7];
assign B16[7]= inB[1] ^ inB[2] ^ inB[4] ^ inB[7];
assign B32[0]= inB[0] ^ inB[1] ^ inB[2] ^ inB[3] ^ inB[4] ^ inB[5] ^ inB[6];
assign B32[1]= inB[1] ^ inB[2] ^ inB[4] ^ inB[7];
assign B32[2]= inB[3] ^ inB[6] ^ inB[7];
assign B32[3]= inB[2] ^ inB[3] ^ inB[4] ^ inB[5] ^ inB[7];
assign B32[4]= inB[3] ^ inB[5] ^ inB[6] ^ inB[7];
assign B32[5]= inB[1] ^ inB[2] ^ inB[4] ^ inB[6];
assign B32[6]= inB[1] ^ inB[2] ^ inB[3] ^ inB[5];
assign B32[7]= inB[1] ^ inB[2] ^ inB[4] ^ inB[5] ^ inB[6] ^ inB[7];
assign B64[0]= inB[0] ^ inB[1] ^ inB[2] ^ inB[3] ^ inB[5] ^ inB[6] ^ inB[7];
assign B64[1]= inB[1] ^ inB[2] ^ inB[4] ^ inB[5] ^ inB[6] ^ inB[7];
assign B64[2]= inB[3] ^ inB[7];
assign B64[3]= inB[1] ^ inB[2] ^ inB[5] ^ inB[6];
assign B64[4]= inB[3] ^ inB[5] ^ inB[6];
assign B64[5]= inB[1] ^ inB[2] ^ inB[3] ^ inB[7];
assign B64[6]= inB[1];
assign B64[7]= inB[1] ^ inB[2] ^ inB[3] ^ inB[5];
assign B128[0]= inB[0] ^ inB[1] ^ inB[3];
assign B128[1]= inB[1] ^ inB[2] ^ inB[3] ^ inB[5];
assign B128[2]= inB[4] ^ inB[5] ^ inB[7];
assign B128[3]= inB[1] ^ inB[3] ^ inB[6] ^ inB[7];
assign B128[4]= inB[3] ^ inB[5];
assign B128[5]= inB[1] ^ inB[5] ^ inB[7];
assign B128[6]= inB[7];
assign B128[7]= inB[1];


GF_mult u0 (.a(B2),.b(B4),.c(B2B4));
GF_mult u1 (.a(B8),.b(B16),.c(B8B16));
GF_mult u2 (.a(B32),.b(B64),.c(B32B64));
GF_mult u3 (.a(B2B4),.b(B8B16),.c(B2B4B8B16));
GF_mult u4 (.a(B32B64),.b(B128),.c(B32B64B128));
GF_mult u5 (.a(B2B4B8B16),.b(B32B64B128),.c(BInvert));

endmodule
```

4.系统软件设计

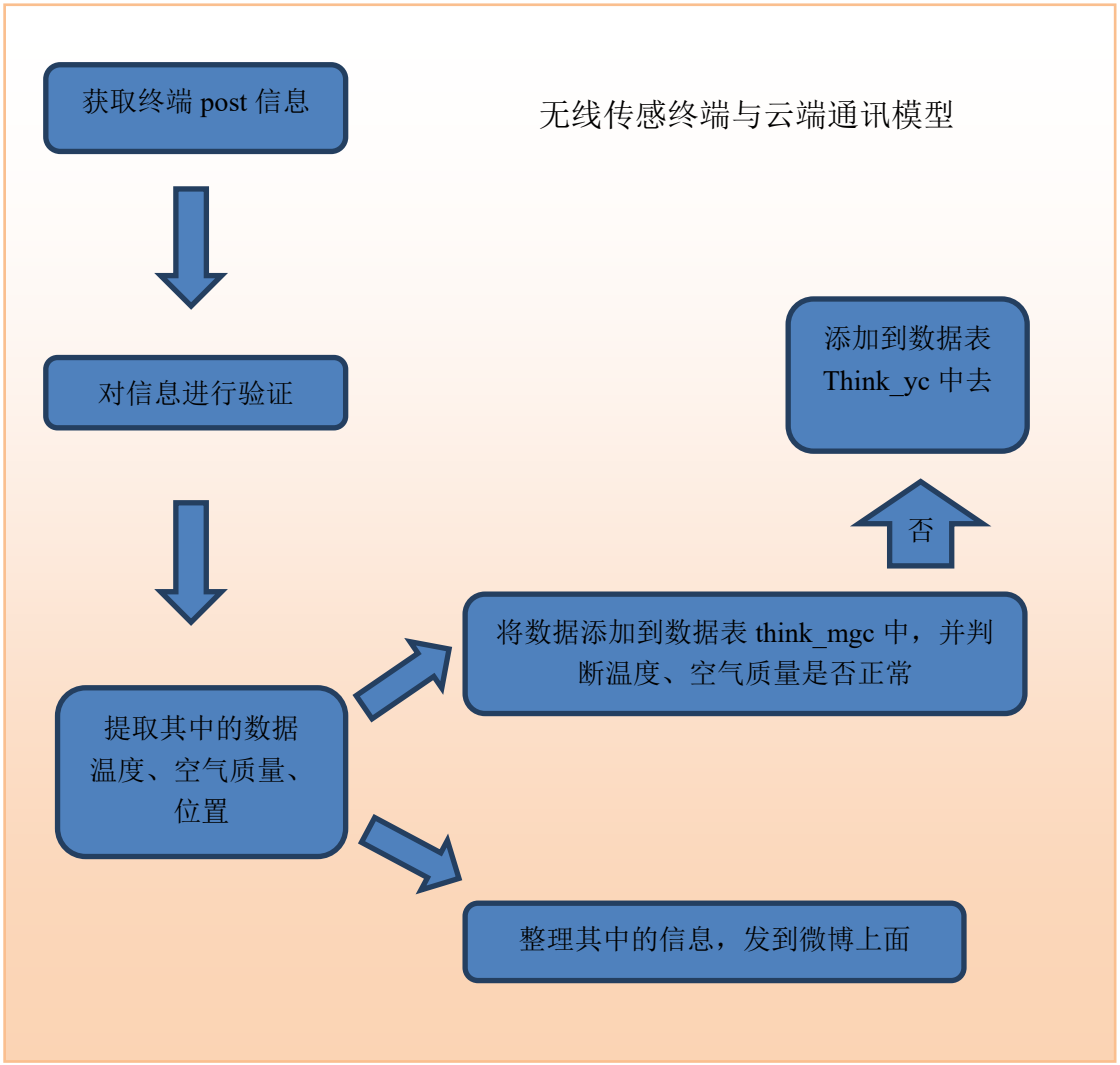
4.1 云端设计

云端主要负责与芯片进行通讯，把芯片提供的信息进行验证，提取其中的温度、空气质量、位置等信息，将其添加到数据库当中保存，方便以后进行分析和制作图表。另外，云端还针对芯片发过来的信息进行整理，将其发表到我们的微博。

微博地址：http://scutiot.sinaapp.com/

在实现的方式上，我们使用新浪 SAE 开发平台搭建我们的服务器。在整个服务器网站的设计上，我们采用了 thinkphp 框架，通过其中的 MVC 分层结构，实现了前台页面，控制

器，数据库连接之间的分层。其中前台负责监测数据的展示和我们项目组的一些简单信息。后台则负责将收集到的信息添加到数据库还有将数据库里面的信息提供给前台去执行。整个流程图如下所示：



4-1 无线传感终端与云端通讯模型

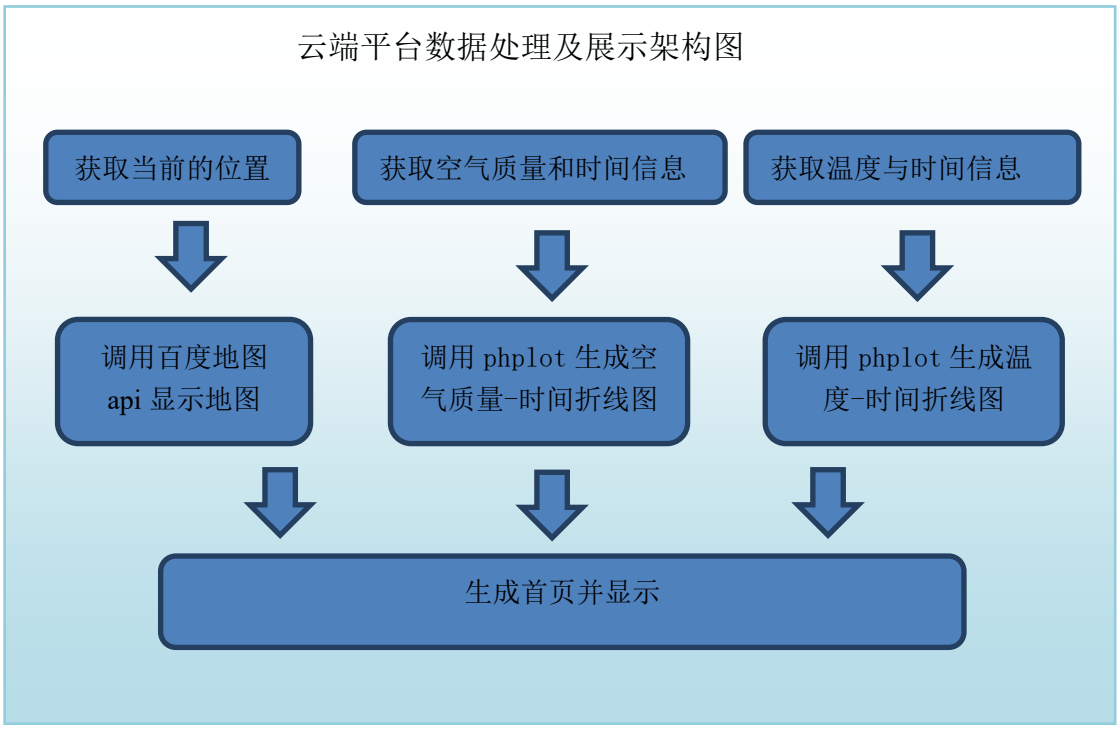


图 4-2 云端平台数据处理及展示架构讯模型

可以看到在整个云端当中，只有一个首页(http://scutiot.sinaapp.com/)，这个首页包含了我们需要在网站前台展现的所有东西，包括图表、位置信息。而后台发微博，保存数据这些都是通过 php 函数来实现的。它们的功能如下：

1. public function index()这个函数主要是操作数据库，提供首页需要的数据，并调用 index.html 把首页展示出来。

2. public function addwd()这个函数主要把芯片传过来的数据进行验证，并加入数据库当中去。

3. public function drawc()这个函数主要生成温度-时间折线图。

4. public function drawd()这个函数主要生成空气质量-时间折线图。

另外除了上面所说的一些功能外，首页还有展示相片、最近的微博显示。这些都给网站提供了很好的互动性。

4.2 无线终端软件实现

无线终端的程序负责温度、空气质量数据的采集，数据的签名以及发送。

无线终端的控制采用实时时钟和中断结合的方式。当无线终端开始工作后，先执行设备控制器初始化（系统初始化、UART 初始化、Timer 初始化、IIC 初始化），然后执行设备和控制器的自检（包括 GPRS 模块的通讯自检以及部分接口的自检），一系列的初始化和自检完成以后，进入数据的的收集和分析处理状态。此时，进入计时器的定时计算，当时钟溢出，中断计数到某个临界值时，进入中断服务函数，打包最近采集到的一系列数据以及位置信息，签名后，通过 HTTP 请求把数据发送到云端并通过 SMTP 服务器发送电子邮件进行提醒，重置计时器。当接收到异常温度或者空气质量数据时，对数据接收的次数进行定时统计，若连续 10 秒（5 次，每次采集数据间隔为 2 秒）采集到异常数据时，则以短信报警的方式发

送报警短信到指定短信平台，和 SMTP 服务器发送报警信息到指定电子邮箱。最后把签名后的报警信息再通过 HTTP 请求向服务器发起报警信息。然后等待一定时间后，重置计时器以及清空报警信息，重新进入数据的收集和分析处理状态。

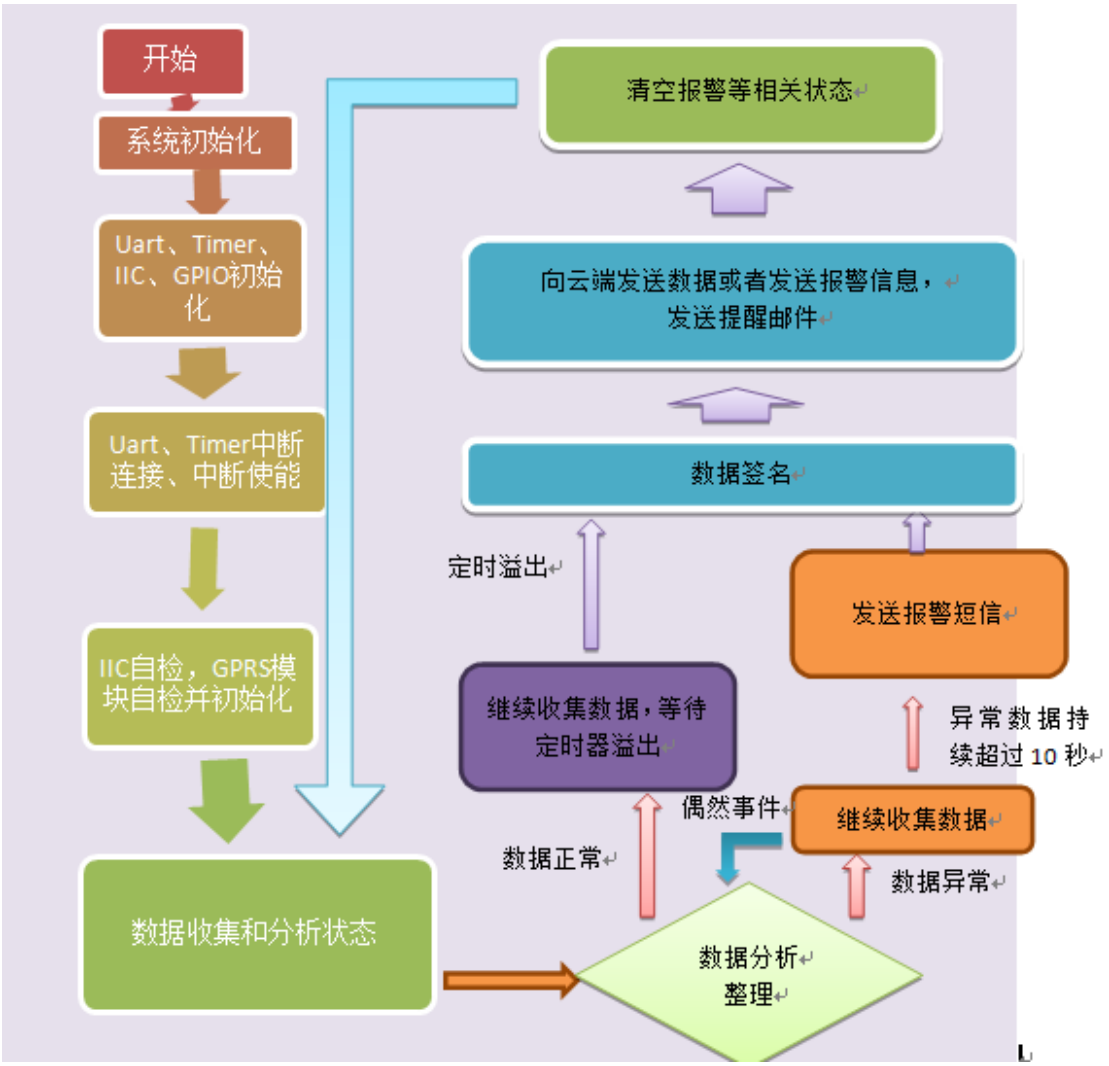


图 4-3 无线终端主程序流程图

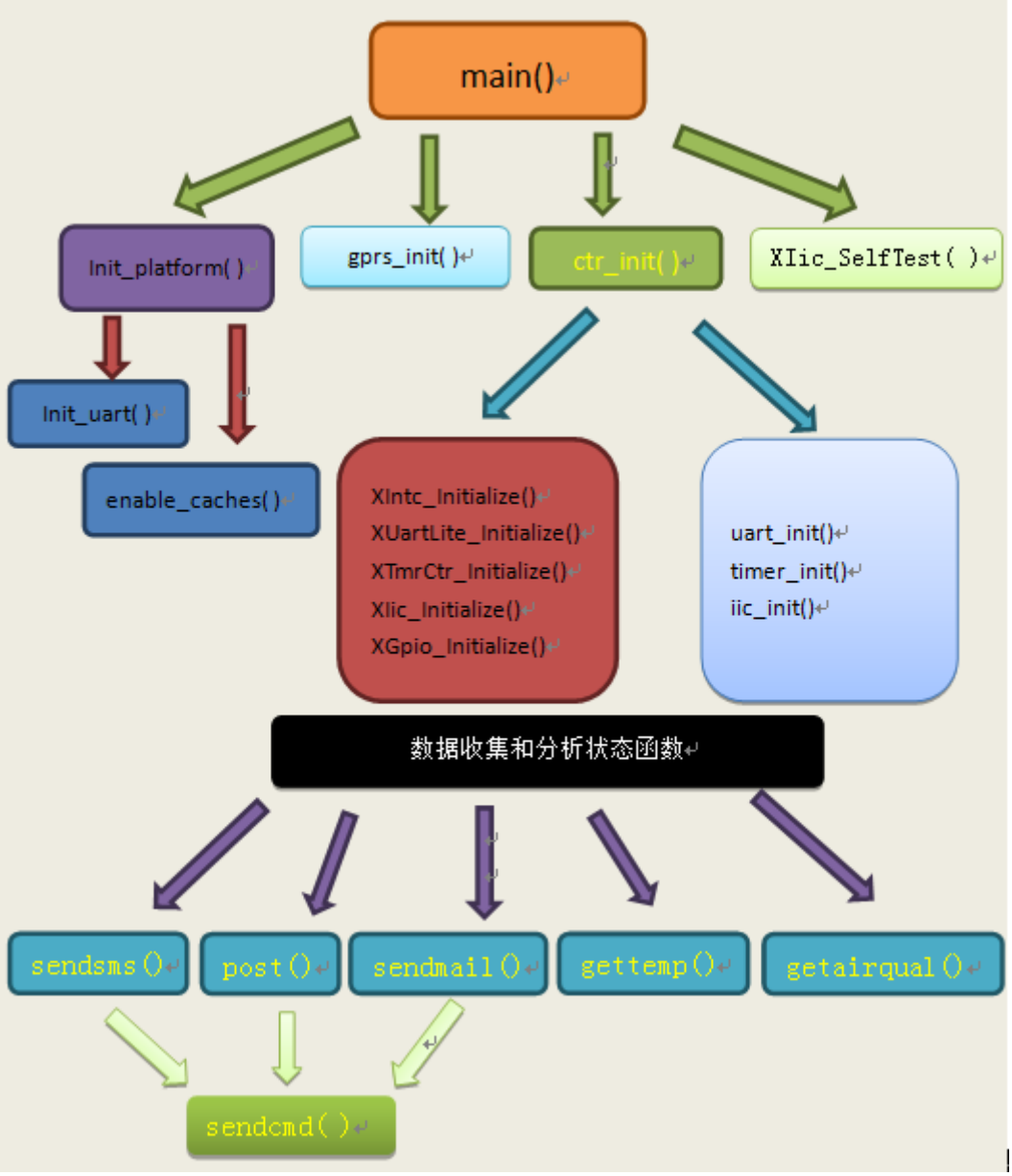


图 4-4 无线终端软件框架

数据的采集分为温度数据的采集和空气质量数据的采集，温度数据的采集需要发送采集数据指令以采集数据，空气质量数据的采集仅仅是获取端口信息就可以完成。该部分包含以下函数：

●intiic_init(XIntc * IntcInstancePtr,XIic *IicInstancePtr,u16 DeviceID, u16 IntrId)函数用于连接 iic 控制器和中断控制器，初始化 iic 通讯,iic 中断使能。

●intgettemp(XIic* InstancePtr,u32 *temp100)函数用于发送指令并获取温度传感器的信息 temp100 是温度的 100 倍。

●intgetairqual(XGpio* InstancePtr,u32 *airqual)函数用于获取空气质量数据 airqual。

数据的发送都是使用 gprs 模块进行发送，包括 email 、post 请求、以及短信。Email 以及 post 请求的实现都是基于 GPRS/GSM 模块本身支持 TCP 协议的基础上，在基本 TCP 连接层上实现了 SMTP 服务以及 HTTP 服务的应用层，该部分包含以下函数：

●intuart_init(XIntc * IntcInstancePtr,XUartLite *UartInstancePtr,u16 DeviceID, u16 IntrId)函数用连接 uart 控制器和中断控制器，初始化 uart 通讯，uart 中断使能。

●intgprs_init(XUartLite *InstancePtr)　GPRS 自检和初始化函数。

●intsendsms(const u8* sms,const u8* phone, XUartLite *InstancePtr)短信发送函数，用于把参数 sms 包含的信息发送到参数 phone 指定的电话号码。

●intsendmail(const u8* content,XUartLite *InstancePtr)电子邮件发送函数，用于把参数 content 里面的内容发送到特定的电子邮箱中去。

●int post(const u8* content,XUartLite *InstancePtr) 云端服务器请求函数，用于把参数 content 里面的内容发送到特定的服务器。

5.系统调试和测试

5.1 调试过程

在编程的过程中，发现错误，然后根据错误的迹象确定程序中错误的确切性质、位置和原因。然后修改设计和代码，以排除错误。排除错误后，必须进行回归测试，防止引入新的错误。

5.2 测试过程

5.1.1 硬件平台的方案测试

我们的团队主要的目的是实现一个距离远、覆盖范围广、数据传输量少的基于无线传感中端的云平台。在温度传感器的使用方案中，我们曾经使用过 DTH11 温湿度度传感器，但是经过实验的结果，发现它的只能是测试物体或者是空气的表面温度，往往是在温度的变化监测上，达不到要求。因此，经过对需求的分析以及对性能指标的指定，我们选取了 MLX-90614 作为我们的温度数据采集传感器。另外，在无线通讯模块的评测与选择当中，我们曾经考虑过使用 WIFI 以及 Zigbee 作为我们的无线通讯工具，但是，经过对性能的分析以及利用智能手机作为实验评估时，发现虽然 Wifi 的传输距离在一定程度上能满足要求，而且速度等性能也很客观，再者，也便于与 Android 终端进行数据交互，但是限于国内 Wifi 的普及程度以及对一些原始森林等环境使用的要求，我们最终选用了 GPRS 模块作为无线传输的终端。

5.1.2 软件平台方案的测试

在项目的早期，我们也曾经考虑过 PC 终端的上位机实现，但是该方案对于越来越趋向于主流的实时信息发布以及信息共享的方式显得有点格格不入，而且我们也曾经开发过 PC 机端上的数据认证以及采集平台，是利用无线终端发数据到 email，然后 PC 机上通过 POP3 协议收信的形式获取数据，但是这种方案最明显的弊端就是不能实时更新信息，从而导致某些敏感信息无法及时接收。因此，我们最终使用了服务器监听无线终端 http 请求的方式以及无线终端短信发送的方式来实现我们的上位机。

5.1.3 彩虹算法方案的测试

随机生成一组数，作为明文输入，作用于签名算法。得到签名算法后，把签名和明文输入到验证算法，查看签名和明文是否匹配；修改明文中的其中一个或多个数值，把

修改后的明文和第一步得出的签名输入到验证算法中，查看是否有提示明文和签名不匹配；修改签名中的其中一个或多个数值，把修改后的签名和第一步得出的明文输入到验证算法中，查看是否有提示明文和签名不匹配。

6.总结

相对计划书，可以说我们的项目的内容发生了比较大的变化，从原来的单一密码芯片设计扩展到了现在的基于无线传感终端的可信云系统的实现。计划书中本来是只有一个无线传感节点的介绍在里面，但是经过我们项目组和指导老师的再三讨论和方案的修改，决定把无线传感节点也一起实现出来，另外又考虑今天广泛流行的云计算确实为我们的工作和生活带来了许许多多的便利，把云计算也融合到了我们的项目中来。与其他的队伍一样，我们的项目同样实现上位机和下位机。但是，与别人的差别是，我们的上位机并不是通过单击的 PC 端实现的，而是通过云端服务器来实现的。但美中不足的是，由于我们缺少相关 FPGA 编程的经验再加上时间的限制问题，对于密码芯片的设计仍然存在部分的 Bug，虽然我们的代码仿真已经通过，并在实际的板级调试中，用实验室的 Virtex5 的开发板也已经通过了逻辑调试，在我们把算法挂载到系统上面的时候，却出现了偶尔的数据出错的问题。因此，我们也正在进行最后的 Bug 修复以及代码的完善。

修改日期： 2012 年 4 月 28 日

经过一段时间的分析以及调试，我们终于把所有的功能都实现了，原来是我们对于 FPGA 的以及硬件逻辑设计的了解不够深入而导致的问题，经分析，原来我们使用 vertex5 的开发板能够通过我们的数据测试是因为我们所设计的密码芯片的性能刚好就在我们 Nexys3 板子的性能临界点，经后期调试以及分析，我们的密码芯片的输入时钟可以达到 80MHZ，仿真过程中，我们使用的是 50MHZ 的时钟作为输入，不到 700 个周期就可以完成一次签名。而在实际的嵌入式系统方案中，出于稳定性的考虑，我们使用的总线时钟是 25MHZ，能够进行稳定的签名。目前，我们与服务器联调的试验也已经完成，结果相当令人满意。

附上我们项目的展示网址: http://scutiot.sinaapp.com/

项目微博发布账号：scutiot 地址： http://weibo.com/u/2684515223

说明：由于前期进行数据测试以及认证部分，所以微博以及网站数据显示中有部分重复数据的出现是手工发送的。

7.参考文献

[1] 刘正东,林宇,曾亮,量子计算机[J] 自然杂志.1998,20(2).

[2] Bernstein, Daniel J. Introduction to post-quantum cryptography. University of Illinois Department of Computer Science Chicago, Springer Berlin Heidelberg,2009

[3] Jintai Ding,Jason E. Gower，Dieter S. Schmidt Multivariate Public Key Cryptosystems [M]. University of Cincinnati USA: Springer,2006.

[4] Matsumoto, Tsutomu and Imai , Hideki (1988). Public quadratic polynomial-tuples for

efficient signature verification and message encryption. In Gucnther, C. G., editor, Advances in cryptology - EUROCRYPT '88, volume 330 of LNCS, pages 419-453. Springer.

[5] Patarin, Jacques (1996b). Hidden Field Equations (HFE) and Isomorphism of Polynomials (IP) : Two new families of asymmetric algorithms. In Maurer, U. , editor, Eurocrypt'96, volume 1070 of LNCS, pages 33-48. Springer. Extended Version: http://www.minrank.org/hfe.pdf .

[6] Moh, Tzuong-Tsieng (1999a). A fast public key system with signature and master key functions. Comm. in Algebra, 27:2207-2222. http://www.usdsi.com/ttm.html .

[7] Patarin,J. The oil and vinegar signature scheme. Dagstuhl Workshop on Cryptography，1997.

[8] Ding, Jintai and Schmidt , Dieter (2005b). Rainbow, a new multivariable polynomial signature scheme. In loannidis, John, Keromytis, Angelos D. , and Yung, Moti , ed-itors. Third International Conference Applied Cryptography and Network Security (ACNS 2005), volume 3531 of LNCS Springer.

[9] 沈晓强. 有限域乘除法研究与实现. 国防科学技术大学 20060301, pp.6

[10] 王进祥 & 毛志刚.GF (2^8) 上快速乘法器及求逆器的设计.微电子学, 1998, 28, 321-324