华南理工大学

South China University of Technology

本科毕业设计（论文）

题目： FPGA 上基于多变量公钥密码体制身份认证系统的设计与实现

学　院　　材料科学与工程学院

专　业　　材料类创新班

学生姓名　　彭峙酿

学生学号　　200932281199

指导教师　　吴为敬

提交日期　　2013　年　5　月　　日

摘　要

关键词

（另起页：外文摘要范例；英文摘要和关键词应该是中文摘要和关键词的翻译）

Abstract

Keyword（Times New Roman 字体，小三号，加粗，居左）: Multivariable system, Predictive control, Environmental test device（Times New Roman 字体，小四号）

（另起页：目录范例）

目　录

（小三号，宋体，加粗，居中，上下空一行，目录由电脑自动生成）

（各章题序及标题：小四号，宋体，加粗，居左；其余用小四号，宋体）

第一章 绪 论

1.1 研究背景

1976 年，W.Diffie 和 Hellman 在《New Directions in Cryptography》中首次提出了非对称密码算法的思想。78 年，Rivest、Adleman 和 Shamir 三位学者提出了安全性依赖于大数因式分解的 RSA 公钥密码算法。由此，密码学的发展产生了一次伟大的革命。公钥密码学的诞生，开拓了密码学领域的许多新的研究领域；比如身份认证、数字签名、密钥管理、分配等。传统的加密解密，只是其一个应用方法。经过三十多年的发展，如今的它已近成为密码学的最主要的研究方向之一。在信息化高速发展的当代，公钥密码学的应用更是无处不在，在信息社会中保护着数以亿记的人的隐私和数据。

但是量子计算机概念的提出，对传统的公钥密码体制构成了严重的威胁。诺贝尔物理学奖得住 Richard Feynman 提出了以量子系统来构成的计算机，其运行速度相对于传统计算机可以大大提高。1995 年 Peter shor 提出了一种基于量子计算机的量子分解算法，该算法利用量子计算机的特性，能够在多项式时间范围内解决离散对数和大数分解等问题。而这些问题正是传统公钥密码体制的核心所在，一旦量子计算机真正问世，那么传统的公钥密码算法如 RSA、ElGamal、ECC 等算法都将遭到攻击。使用这些密码算法的银行、政府、国家将会无一幸免。这就为公钥密码学体制提出了一个新的要求：构造能够抵御来自量子计算机攻击的密码学算法。由此，多变量公钥密码体制应运而生。

在有限域上解决多变量高次方程组问题，被人们称为 MQ 问题。MQ 问题已经被证

明是 NP 困难问题，且量子计算机也无法在多项式时间范围内解决 MQ 问题，因此他能够作为设计密码学的数学基础。目前已经有很多基于 MQ 问题所提出的对称密码算法和非对称密码算法。在对称密码算法中，BerBain 等人提出了 QUAD 流密码，它的安全性可以证明能够规约到 MQ 问题。在非对称密码学中，即公钥密码学中，已经有很多公钥密码方案被提出。如 Matsumoto-Imai 公钥密码体制、Oil-Vinegar 公钥密码体制等等。基于 MQ 问题的密码学方案被人们统称为多变量公钥密码体制。多变量公钥密码体制相对于传统的具有运算速度快的优点，所以他能够很好得在资源有限的低端设备：如 IC 卡、无线传感网络中应用。故多变量公钥密码体制在低端设备上的应用，也是目前全世界的一个研究趋势。

可编程逻辑门阵列（FPGA）是在可编程阵列逻辑、通用阵列逻辑、可擦出的可编程逻辑等基础之上发展出的新产物，他作为专用集成电路领域中的一种半制定电路，克服原先可编程逻辑器件有限的门电路资源和不便，可以完成更复杂的逻辑电路。可作为多变量公钥密码体制在低端设备上实现的桥梁。

目前现存多变量公钥密码体制的安全性不仅依赖于 MQ 问题，同时也依赖于多项式同构问题，即 IP 问题，并且部多变量公钥密码方案已经被证明是不安全的。由此，在多变量公钥密码体系中，设计一种安全性仅依赖于 MQ 问题、并能在低端设备中实际运用的安全的密码学方案十分重要。

1.2 国内外研究情况

传统的密码学算法如 RSA、ECC 等他们的安全性都依赖于一些被认为是难解的数学问题如因式分解问题、离散对数问题等，一旦某个基础的数学问题被攻破，那么基于该问题的密码学方案也将被攻破。1994 年 Peter Shor 提出了在量子计算机上进行因式分解的算法，该算法从数学理论上设计一系列酉变换，利用量子计算机的并发性能够在非常短的时间内完成在传统计算机上被认为是计算不可行的因式分解问题，。物理学家以证明，一切的酉变换都是可以实现的门电路，即若量子计算机问世，Shor 的方案是可行的。传统的密码学体制如 RSA 等在量子计算机面前遭到了严重的威胁。

由此密码学家门便开始寻找替代因式分解问题的一些其他安全的数学问题，且要求这些问题能够抵抗量子计算的攻击。在有限域上解决多变量二次方程组的问题，被人们称为 MQ 问题。Garey 已经在 1979 年证明了 MQ 问题在有限域 $GF(2)$ 上是一个 NP 完全问题，而 Patarin 又在 1997 年证明了在任意有限域上 MQ 问题都是 NP 完全问题。一个随机的 MQ 问题实例都被认为是难解的。相对于密码学中传统的因式分解或离散对数问题，目前并没有已知多项式时间算法能在量子计算机上解决 MQ 问题。这使 MQ 问题能够被用作 RSA 等传统密码学算法的替代品。一个包含了多个变量二次方程组的函数，我们成为 MQ 函数。MQ 函数能够被用作单向函数使用，并且它有用很短的输入和输出。

攻击 MQ 问题的最常见方法是 Grobner 基方法，该方法无在时间上还是空间上都是指数级的。

目前已经有很多基于 MQ 函数所提出的对称密码算法和非对称密码算法。在对称密码算法中，比较出名的是 BerBain 等人提出了 QUAD 流密码。QUAD 是一个实用且可证明安全的流密码方案。该方案依赖于 MQ 方程的不断迭代生成密钥流，它的安全性可证明可以规约到 MQ 问题的难解性。在非对称密码学中有许多公钥方案已经被提出，这些方案被统称为 MPKC。早起比较出名的 MPKC 密码方案是 MI 密码体制，它是由 Matsumoto 和 Imai 于 1998 年提出的，但是该方案后来被证明存在线性攻击方法，后来又提出了 HFE、STS、UOV 等方案。但是这些 MPKC 现存方案的安全性不仅依赖于 MQ 问题，同时也依赖于多项式同构问题，即 IP 问题。IP 问题是要在两组多变量方程组之间寻找一个特殊的变换。许多关于多项式同构问题的密码学分析已经提出。事实上，一些 MPKC 的密码学方案已经被证明是不安全的。

2011 年 Kitashinagawa 等人在 Crypto 上第一次提出了一种基于 MQ 问题的公钥身份认证方案。假设存在一个非交互性的承诺方案，并且该方案具有统计隐藏性和计算绑定性。该公钥身份认证方案并不依赖于多项式同构问题，而是直接依赖于 MQ 问题的困难性。同时该对于承诺方案的假设是很自然的，因为他可以由一个抗碰撞的单项 hash 函数构成。该身份认证协议对于 MQ 问题的求解而言，是统计零知识的知识论证。基于 MQ 问题的难解性，该身份认证方案包括协议的顺序构造方法和并行构造方法，并且在被动攻击和主动攻击下都是可证明安全的，并且他的安全级别与其他知名认证方案相同。

1.3 实验平台概述

本设计主要对 kitashinagawa 提出基于 MQ 问题的身份认证体制进行实现并改进。并将其客户端在 FPGA 上实现。服务端在计算机系统中用 java 语言实现。本人在 FPGA 平台上实现了高速并行化的身份认证系统。我们选择基 Spartan-6FPGA 的数字系统开发平台的 nexys3 开发板进行仿真和试验。开发环境是 xilinx ise 13.2。并通过 xilinx 自带工具对电路进行面积、功耗分析。

1.3.1 FPGA 介绍

FPGA 即现场可编程门阵列它是在传统用的额可编程器件的基础上发展出来的产物。FPGA 设计能够作为为专业集成电路的半制定电路，在功能上有许多优点。它能够客户传统的可编程器件的限制，由能够克服 ASIC 的不足。通过硬件描述语言完成电路设计之后，FPGA 设计者可将通过集成工具快速的将程序写入到 FPGA 上进行测试，而且它能够多次反复使用。这极大的加速了 IC 行业的研发时间、降低了研发成本。目前 FPGA 以成为 IC 开发必不可少的一种工具。

1.3.2 Xilinx ISE 介绍

本文中使用的 FPGA 开发平台为 Xilinx ISE 13.2，该软件为 FPGA 硬件设计工程一

个整体的开发换进，包括图形或文本输入、综合工具、实现工具、下载工具等。本次设计本人使用了 Xilinx ISE 作为开发的环境，使用 verilog 硬件描述语言进行硬件设计。借助 xilinx 的开发环境，我们可以很轻松的完成 RTL 到比特流的设计流程，利用其中集成的工具，同时可以很清晰的分析设计的关键逻辑，并且能够生成详细的功耗情况和资源使用情况，有助于我们对设计进行改善。本文中我们使用了 ModelSim 作为仿真软件，Modelsim 具有仿真速度快、结果精确等优点，能够有助于程序的验证和问题的发现。

1.3.3 nexys3 开发板介绍

本次试验中我们使用 nexys3 开发板作为实验板，Nexys3 FPGA 开发板是 Spartan-6 FPGA 家族的一员。Spartan-6 相比于旧的产品，提升了 50%的容量，并且拥有更好的性能。以 Xilinx Spartan6 XC6LX16-CS324 为其编程芯片，同时拥有丰富的片外资源。nexys3 提供了 32Mbytes 的相变非易失性存储器，16Mbytes 的 RAM、USB 通讯口、VGA 口等资源。这让 nexys3 成为众多数字设计系统的理想宿主，包括 xilinx 的 MicroBlaze 嵌入式处理器等。nexys3 与 xilinx 的 CAD 工具完全兼容。同时它使用 Digilent 的最新 Adept USB2 系统提供 FPGA 和 ROM 编程。

1.4 研究内容

本文主要中作集中于对基于 MQ 问题的身份认证方案的 FPGA 实现，并对方案向数字签名和基于身份签名方向进行扩展，主要成果如下。

（1） 实现并优化了 FPGA 上 MQ 函数运行程序。

（2） 实现并优化了 FPGA 上真随机数生成器。

（3） 是在并优化了身份认证方案控制器。

（4） 对该身份认证方案进行数学扩展，得到安全的数字签名和基于身份的数字签名。

1.5 论文的组织结构

本文结构安排如下：

第一章：绪论。主要介绍本文的研究背景、研究内容和研究环境。并总结了本文的研究成果。

第二章：基础知识介绍。主要介绍了有限域、MQ 问题、交互证明和零知识证明、证据不可区分性和证据隐藏性、承诺函数、身份认证等基本概念。

第三章：FPGA 上实现基于 MQ 问题的身份认证系统。首先介绍了基于 MQ 问题的身份认证的数学方案，并将该方案实例化后设计了一套身份认证系统，并将其在 FPGA 上实现。

第四章：分析了算法的性能和资源使用情况，并提出改进方向。

第五章：对原算法的数学方案进行扩展，提出将其改成数字签名方案和基于身份的数字签名方案的方法。

最后是本文的结论部分，总结了本文所取得的成果，并提出了改进方向。

第二章　基础知识介绍

2.1 有限域

定义 2.1(域) 对于一个非空集合 F，定义在其上的两种二元运算为乘法运算$\bullet$和加法运算$+$。如果满足以下条件，我们称集合 F 关于二元运算$+$和$\bullet$构成一个域$\{F,+,\bullet\}$。

(1) 加法和乘法满足交换律:对任意$\alpha,\beta\in F$ ，都有

$$\alpha+\beta=\beta+\alpha$$

$$\alpha\bullet\beta=\beta\bullet\alpha$$

(2) 加法和乘法满足结合律:对任意$\alpha,\beta,\delta\in F$,都有

$$(\alpha+\beta)+\delta=\alpha+(\beta+\delta)$$

$$(\alpha\bullet\beta)\bullet\delta=\alpha\bullet(\beta\bullet\delta)$$

(3) 加法对乘法满足分配率：对任意$\alpha,\beta,\delta\in F$,都有

$$(\alpha+\beta)\bullet\delta=\alpha\bullet\delta+\beta\bullet\delta$$

(4) 加法存在零元0 ，乘法存在单位元1 ：对任意$\alpha\in F$

$$\alpha\bullet 1=\alpha$$

$$\alpha+0=\alpha$$

(5) 任意元素都存在加法逆元素：对任意$\alpha\in F$

存在$\beta\in F$ 使得$\alpha+\beta=0$

(6) 任意非零元素都存在乘法逆元素：对任意$\alpha\in F$

存在$\beta\in F$ 使得$\alpha\bullet\beta=0$

定义 2.2（有限域）如果一个域 F 中只存在有限个元素，那么我们称域 F 为有限域。有限域又被称为伽罗瓦域。若有限域中有 q 个元素，那么我们可以将其简写为$GF(q)$ 。

2.2 有限域上的 MQ 问题

定义 2.3 MQ 问题 给定有限域$GF(q)$上的一组非线性多变量方程组Q，其中$Q=\{Q_1(x),...,Q_m(x)\}$。求解$x=\{x_1,...,x_n\}\in GF(q)$，使其满足$Q_i(x)=0$的问题为 MQ 问题。

MQ 问题中的方程组可以表示为：

$$Q_1(x_1,...,x_n)=\sum_{1\le i\le j\le n}\alpha_{1ij}x_ix_j+\sum_{1\le i\le n}\beta_{1i}x_i+\lambda_1$$

$$\vdots$$

$$Q_n(x_1,...,x_n)=\sum_{1\le i\le j\le n}\alpha_{nij}x_ix_j+\sum_{1\le i\le n}\beta_{ni}x_i+\lambda_n$$

目前有限域上 MQ 问题已经被证明对于任意有限域$GF(q)$都是 NP 困难问题，即不存在有效的多项式时间算法能够求解该问题。目前所有解决该问题的方法都是指数级复杂度的。最常见的 Grboner 基攻击方法无论在时间上还是在空间上都是指数级的。Bouillaguet 等人指出如果 m=n<200 的话，无法实现深度搜索。他们提出了一个改进的深度搜索算法来攻击 MQ 方程，被认为是目前最好的攻击方法。目前所知，在二元域$GF(2)$中对 84-bits 输入，80-bits 输出的 MQ 方程的攻击需要$2^{88.7}$ ($>2^{80}$) 个操作。所以 MQ 问题能够被广泛应用于密码学方案之中。

以下我们将有 n 个未知数、m 条方程、定义在有限域$GF(q)$上的 MQ 方程简写为$MQ(n,m,F_q)$。

定义 2.4 （原像稳固） 我们说$MQ(n,m,F_q)$是原像稳固的：如果不存在多项式时间算法，输入(F,v)，$F\in_R MQ(n,m,F_q)$，$v\in_R F_{mq}$，能够以不可忽略的概率能够找到一个$s\in F_{nq}$满足 $F(s)=v$。

定义 2.5 （二像稳固） 我们说$MQ(n,m,F_q)$是二像稳固的：如果不存在多项式时间算法，输入(F,x)，$F\in_R MQ(n,m,F_q)$，$x\in_R F_{nq}$，能够以不可忽略的概率找到一个第二像$x'\in_R F_{nq}(x'!=x)$,满足$F(x)=F(x')$。

2.3 交互证明与零知识证明

定义2.6（语言与证据）令R是一个二元关系，定义语言$L(R)=\{x\mid \exists w使(x,w)\in R\}$。则$w$是$x\in L(R)$的证据。

定义 2.7 （交互证明） 交互证明是一个至少两方参与的协议，包括证明者 P 和验证者 V。对于语言 L，定义在其上的交互证明(P,V)需要满足如下条件：

（1）对于任意属于 L 的语言，V 都一定接收。

（2）P 和 V 进行交互能够在多项式时间范围内完成。

（3）对于任意不属于 L 的语言，任意计算能力的模仿者 P' 使 V 接收的概率都是可忽略的。

交互论证的概念来源于交互证明，其定义也基本一致。但是他对要求（3）放宽条件，只需要对多项式计算能力的证明者满足即可。

零知识证明是 2010 年图灵奖获得者 goldwasser 等人在上个世纪 80 年代提出的一个系统。在零知识证明系统中，证明者 P 能够在不向验证者 V 提供任何有用信息的前提下使 V 相信某个论断是正确的。

零知识证明的一个例子：P 要向 V 证明他是某个房间的主人，即证明自己拥有房间的钥匙。P 有几种方式可以选择：

1） P 可以选择将房间的钥匙直接给 V，让 V 来打开房间门。

2） P 可以在 V 面前用钥匙打开房间门。

3） P 让 V 确认房间里的某一样东西，然后 P 将他拿出来。

上面 3 中方案都能够证明 P 拥有房间的钥匙，但是前面两种方案均有一定概率泄露与钥匙相关的秘密。第三种方案是零知识证明，因为 P 在没有向 V 任何关于钥匙的信息的前提下，证明了自己对房间的拥有权。

定义 2.8 （零知识证明）

定义在语言 L 上的交互证明或论证(P,V),对任意 $(x,w)\in R$。我们有以下几种零知识证明或论证:

完美零知识：能够构造一个能够产生与真实交互副本具有相同分布的模拟器。

统计零知识：剔除掉一些造成错误的情况，能够构造出一个与正式交互副本具相同分布的模拟器。

计算零知识：能够构造出一个在计算上无法区分真实交互副本与模拟器输出的模拟器。

无用零知识：即不存在一个以上的模拟器，但是我们可以证明在交互中 V 无法从 P 处获得任何有用的信息。

零知识证明还有顺序零知识证明、并行零知识证明、交互式零知识证明等系统。

2.4 证据不可区分和证据隐藏

零知识证明协议在一些特殊的构造下面有时候很难继续保持零知识性，这给其应用抹上了阴影。在这样的基础下，Feige 和 Shamir 等人提出了证据不可区分性和证据隐藏

性在很多密码学协议中来代替零知识性。

在一个交互式证明协议（P，V）中，如果 P 使用多个秘密证据中的一个来证明一个断言，而且 V 不能够判断 A 使用了哪一个证据。那么我们说，这个协议是证据不可区分的。

定义 2.9 （证据不可区分） 证明系统（P，V）对于 R 是证据不可区分的：如果任意算法V'与 P 进行交互 ，当公共输入为x 时。$w_1, w_2 \in w(x)$。视$V'_{P(x,w_1)}(x)$与视$V'_{P(x,w_2)}(x)$都不可区分。

在一个交互式证明协议（P，V）中，如果 V 在证明结束后他不能够计算出任何在协议开始前他并不知道的新证据。那么我们说，这个协议是证据隐藏的。证据隐藏是信息安全中的一个自然的安全要求，他能够在很多方案中代替零知识证明协议。

定义 2.10 （证据隐藏）证明系统（P，V）对于 R 是证据隐藏的：如果存在一个期望多项式时间的证据提取器 M 对任意不均匀多项式时间算法V' 。都有$\Pr[V'_{P(x,w)}(x) \in w(x)] < \Pr[M(x;V',G) \in w(x)] + v(n)$ 。其中 G 是一个安全的生成器，生成$x = G(1^n)$ 且$\Pr[(x, C(x)) \in R] < v(n)$对任意算法 C 都成立。

如果拿零知识性和证据隐藏性相比较，可知。零知识性保证了在执行一个协议时没有任何信息被泄露。而证据隐藏性是要保证 P 所有用的证据不被泄露，对于其他非证据的信息并没有进行说明。显然，在密码学方案中，人们会更喜欢使用零知识协议，但是不行的是零知识协议在一些不同的构造方式下并不能保证零知识性，因此限制了他的应用。而证据不可区分性是在任何构造下都能够保持的，所以人们能够将证据不可区分性用到任何密码学协议中，并且保持该协议的特定性质，包括证据隐藏性。

为了能够在密码学协议中使用证据不可区分性，那么该协议必须是非平凡的（如果协议的证据集合$w(x)$ 只拥有一个证据，那么这个协议是平凡的协议）。以下给出一些结论：

如果一个协议证据不可区分的，那么他在任意构造下都能保持证据不可区分性。

如果一个协议是证据不可区分的，并且证据关系是由无爪函数生成的，那么这个协议一定是证据隐藏的。

如果一个协议是零知识的，那么这个协议是证据不可区分的。因此如果我们使用一个零知识的协议，并行重复该步骤 n 次后所得的新协议也许不是零知识的，但是它一定是证据不可区分的。

如果单项函数存在，任何 NP 语言都存在一个常熟轮的证据不可区分证明系统。

2.5 承诺方案

一个字符串承诺方案可表示为 Com.简单的说，一个承诺方案是一个有至少两方参

与的两个阶段的协议。参与方被称为承诺者与接受者。通过这个协议，承诺者能够实现自己与一个数字绑定。如果承诺的是一个字符串，那个这个承诺方案可以被称为字符串承诺方案。在第一阶段，承诺者能够实现自己与一个字符串相绑定，并将它发给接受者。接受者此时并不能打开承诺。即接受者无法获得有关承诺者所承诺的任何信息。在协议的第二阶段，接受者能够获得一个承诺者所承诺值。第一个阶段称为承诺阶段，第二个阶段称为承诺解释阶段。我们要求该承诺方案方案具有统计隐藏性和计算绑定性。正式的说，前者意味着，在第一阶段结束时，没有任何接受者能够区分两个不同字符串的承诺值，即使接收者在计算能力上没有限制。后者意味着具有多项式能力的发送者无法改变承诺的字符串。这样的承诺方案能够从抗碰撞的 hash 函数中构造，并且该交互式的承诺方案能够从任意单向函数构造，也就是说该承诺方案能够由 MQ 函数构造。

可以用一个简单的故事形象的描述承诺方案：A 将一封承诺书装在加锁的盒子里交给 B 保管，B 在 A 打开锁之前无法看到承诺书的内容。而 A 最后再打开锁的时候，也无法改变承诺书的内容。

计算绑定性:我们强调承诺方案的计算绑定性，是指保证任意多项式时间的发送者只能以一种方式打开承诺，承诺值与某个特定的字符串是绑定的。承诺者无法找多项式时间范围内找到第二个满足承诺条件的字符串。所以一旦承诺者做出承诺，他就无法抵赖。

统计隐藏性：计算绑定性只能抵挡多项式时间的发送者，隐藏性则是信息论意义上的安全的。即对不同的字符串，承诺函数获得的承诺值的分布是相同的。任意一计算能力无穷的接受者在承诺揭开前是无法知道承诺字符串的值。

2.6 身份认证

一些身份认证方案是一个涉及 P，V 的协议。他允许 P 向 V 证明他的身份，并且在多项式次数范围内，V 无法在其他人面前模仿成 P。

定义 2.11 （身份认证）对于一个身份认证协议（*P,V*）（P,V）它应该满足如下条件:

（1） 完备性:对每个合法的用户，验证者接它的概率都应该为 1.

（2） 安全性：当一个多项式能力非法用户模仿一个合法用户执行（P,V）协时，V 接受他的概率是可忽略的。

最简单的，我们可以通过一个单向函数来构造身份认证系统。如果单向函数是存在的话，那么身份认证机制也是存在的。

使用证据隐藏的知识证明来构建身份认证协议时非常理想的方案。因为证据隐藏的身份认证协议中，P 的证据是不会泄露的。那么就没有人能够模仿 P，除非他本来就知道 P 的证据。

身份认证方案的形式化定义：一个身份认证方案可由四个多项式时间算法组成。

(*Setup,Gen,P,V*）

Setup 是初始算法，其输入为安全参数，输出系统参数 param。

Gen 是秘钥生成函数，其输入为系统参数 param，输出为公钥和私钥对（pk, sk）。

证明者 P 和验证者 V 是交互式协议，他们的共同输入是（$param, pk$）。对于 P 而言，密钥 sk 也是他的一个秘密输入，只有他自己知道。通过多轮交互，V 最后输出一个认证结果来判定 P 的真实身份。协议（P,V）被称作认证协议。

对于身份认证协议的攻击：主要考虑敌手的目标是要在不知道秘钥的情况下模仿证明者 P。这里介绍两种攻击模式，分别是主动攻击和被动攻击模式。在被动攻击模式下，敌手能够获取真正的证明者 P 和诚实的验证者 P 的通讯副本。在主动攻击模式下，敌手能够与证明者交互。主动攻击下的安全性的要求要强于被动攻击下的安全性要求。

第三章 FPGA 上实现基于 MQ 问题的身份认证系统

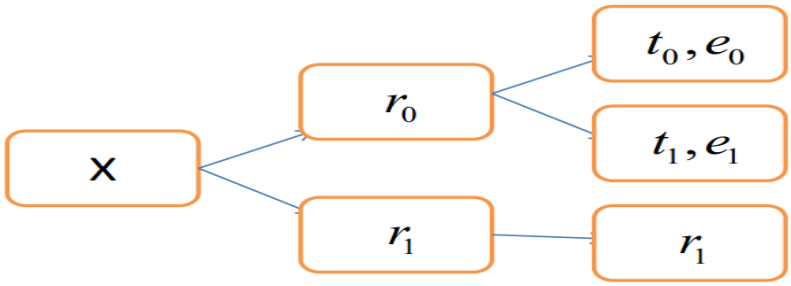
3.1 基于 MQ 问题的身份认证方案

3.1.1 总体描述

2011 年 Kitashinagawa 等人在 Crypto 上第一次提出了一种基于 MQ 问题的公钥身份认证方案。在实际参数选择时，该方案的公钥大小、秘钥大小、通讯量与已知的方案都是相当的。在 80bits 的安全要求下，3 步认证协议的公钥和秘钥甚至只有 80bits 和 84bits 长度，相对于已知的身份认证方案小了很多。在 3 步通讯协议中，当模仿的概率低于 2^{-30} 时，通讯量为 29640bits。该方案使用了具有压缩功能的 MQ 函数（160bits 到 80bits），尽管秘钥大小和通讯量大小会随着实际参数增大。在这种情况下，这样的 MQ 函数是原像稳固的，本方案即使在并行构造时也是抗主动攻击的。

构造技术：零知识证明通常都采用分割技术。证明者首先将他的秘密分成若干份子秘密，然后在不透露秘密本身的情况下，向验证者证明一些份额的正确性。如同指数函数 $x \to g^x$ 或线性函数 $x \to Mx$ 的一些同构性质在这个方法上十分有用，因为我们可以简单的将秘密切分 $s = r_0 + r_1$。那么多对应的值就可以变成 $g^s = g^{r_0} g^{r_1}$ 和 $M_s = Mr_0 + Mr_1$ 。但是 MQ 方程 $(x_1,\ldots,x_n) \to (y_1,\ldots,y_n)$ 看上去并没有这样的性质。因此这里引入了一个新的分割技术。这个分割技术使用了 MQ 函数的差分形式的双极性。定义 MQ 方程 F 差分形式 G 是一个方程。满足 $G(x_1,x_2) = F(x_1+x_2) - F(x_1) - F(x_2)$. 通过简单计算可以证明，该函数具有双线性，这样就可以依据这种双线性性构造新的分割技术。函数 G 是被广泛适用于多变量公钥密码算法的密码学分析。Kitashinagawa 首次将应用于构建公钥身份认证方案。

该分割技术的简单描述如下：定义s 和$v=F(s)$ ，s 为私钥，v 为公钥。当私钥被分为$s=r_0+r_1$ ，那么公钥就被分割为$v=F(r_0+r_1)$ ，根据方程$G(x_1,x_2)=F(x_1+x_2)-F(x_1)-F(x_2)$. 通过应用差分函数 G，可得$v=F(r_0)+f(r_1)+G(r_0,r_1)$ 。继续吧r_0 和$F(r_0)$ 拆分成$r_0=t_0+t_1,F(r_0)=e_0+e_1$ 。相应的，公钥就被分成$v=(G(t_0,r_1)+e_0)+(F(r_1)+G(t_1,r_1)+e_1)$ 两部分。每一部分都可以用三元组(r_1,t_0,e_0) 或者三元组(r_1,t_1,e_1) 来表示。没有关于秘钥的信息能够从其中任何一个中获取。



通过简单计算可以验证，函数G是双线性的。G的表达式可以写成$G=(g_1,.....,g_m)$，

$g_l(x,y)=\sum_{i,j}a_{l,i,j}(y_ix_j+x_iy_j)$

3.1.2 三步认证协议

这里构造一个身份认证协议，定义在 R 上具有错误率为 2/3 的 3 步统计零知识论证。

Setup 初始算法： 令λ 为安全参数，$n=n(\lambda)$，$m=m(\lambda),q=q(\lambda)$ 是多项式界函数。初始算法输入1^λ 并输出 m 个包含 n 个相同随机多变量二次方程。系统参数$F\in_R MQ(n,m,F_q)$.

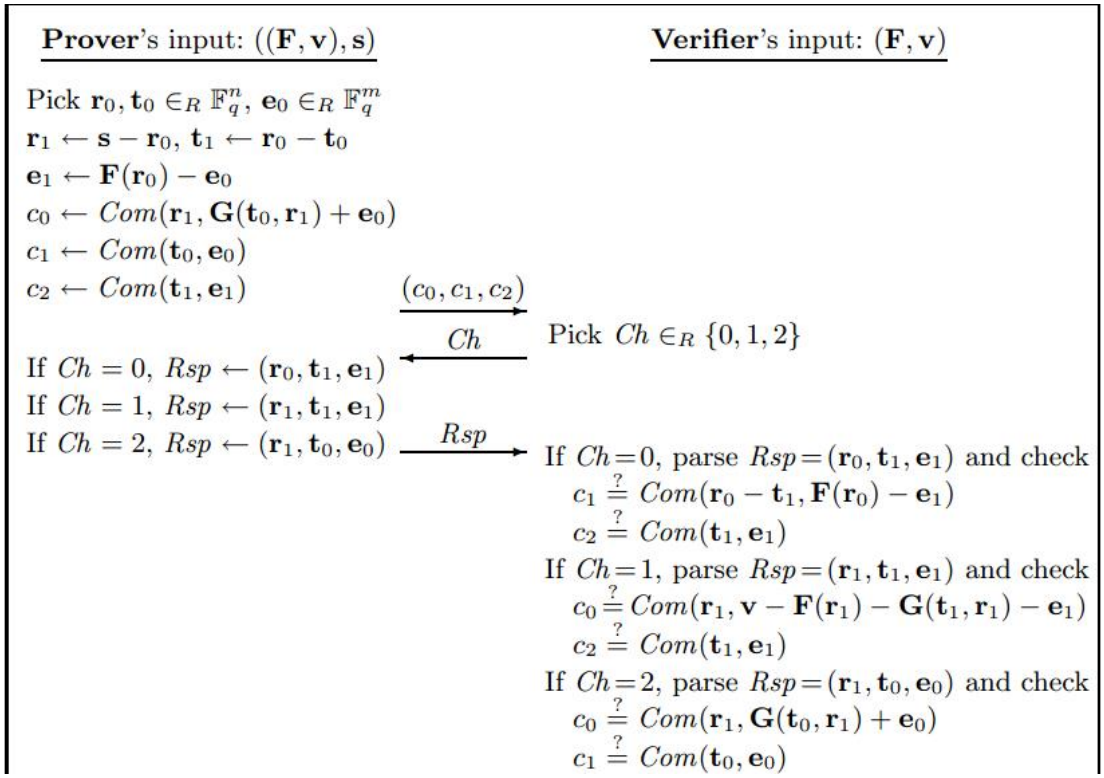
秘钥生成算法： F 为输入，随机选择向量$s\in_R F_{nq}$ 为秘钥，并计算公钥$v=F(s)$ ，输出公钥与秘钥对(pk,sk)=(v,s)作为某个证明者 P 公私钥对。

身份认证协议：3 步协议的基本思路是证明者 P 向验证者 V 证明他有用满足等式 v=F(s)的密钥 s。通过选择分割技术，我们可以将其转化为证明拥有满足下列两个等式。

1） $G(t_0,r_1)+e_0=v-F(r_1)-G(t_1,r_1)-e_1$

2） $1)(t_0,e_0)=(r_0-t_1,F(r_0)-e_1)$

的数据 $(r_0,r_1,t_0,t_1,e_0,e_1)$。如果该数据组满足 1、2 两式，那么 $v=F(r_0+r_1)$ 。则可证明证明者知道秘钥 s，由此证明自己的身份。 根据验证者提供的挑战 $Ch\in\{0,1,2\}$ ，证明者提供 $(r_0,t_1,e_1),(r_1,t_1,e_1)$ 中的一个。 r_0,r_1,t_0,t_1,e_0,e_1 是由前面定义的切分技术产生的。r_0,t_0,e_0 是随机选取的每次都不同，所以验证者从这三个数据组组的任意一个中无法得到任何关于 s 的信息。



三步认证协议如图，为了简单起见，com 函数也有一个随机字符串输入 p，但是图中没有专门写出来。证明者 P 首先随机选取 $r_0,t_0\in_R F_q^n$ $e_0\in_R F_q^m$。然后计算 r_1,t_1,e_1 。得到 6 元组 r_0,r_1,t_0,t_1,e_0,e_1。并通过承诺函数 Com 计算承诺 c_0,c_1,c_2 发给验证者 V。验证者受到承诺 (c_0,c_1,c_2) 后，随机选择挑战 $Ch\in\{0,1,2\}$ 并将该挑战值发送给 P。P 收到挑战之后，根据挑战返回三元组给 V。V 在根据挑战和三元组进行等式验证。若两个等式都满足，则 V 输出 1，否而输出 0.这可以用 $Dec(F,v;(c_0,c_1,c_2),ch,Rsp)$ 。可以很容易看出，如果如果证明着确实拥有秘钥 s，那么验证者总会接受他。因此，3 步认证协议拥有完备性。

3.1.2 安全性

定理 3.1 如果承诺方案 com 是统计隐藏的，那么该三步认证协议是统计零知识的。

证明:令 S 是输入为 F 和 v 的模拟器，但是该 S 并不知道秘钥 s。该模拟器与一个作弊的验证者 CV 进行交互。模拟器能以 2/3 的概率模仿诚实的验证者 P。模拟器 S 随机

得选择一个值 $Ch^* \in R\{0,1,2\}$ ，向量 $r_s', r_0', t_0' \in_R F_q^n, e_0 \in_R F_q^m, Ch^*$ 是预测作弊者 CV 不会选择的 Ch。那么通过构造，我们可以得到，当 $Ch^* != Ch$ 时，模拟器 S 能够通过 CV 的验证。通过构建一个黑盒模拟器 S，并且以 2/3 的概率输出正确的副本。并且，S 输出的分布与真实副本的分数在统计上是不可区分的，所以我们说，三步认证协议是统计零知识的。

定理 3.2 如果 Com 函数是计算绑定的，那么该三步认证协议是 R 上知识错误率为 2/3 的知识论证

证明：令 $((c_0,c_1,c_2),ch_0,Rsp_0)$ ，$(c_0,c_1,c_2),Ch_1,Rsp_1)$ 和 (c_0,c_1,c_2,Ch_2,Rsp_2) 为 $Ch_i =_i$ ， $Dec(F,v;(c_0,c_1,c_2),ch_i,Rsp)=1$ 对任意 $i \in \{0,1,2\}$.那么通过使用者三个副本，可以知道我们要么能够攻破承诺协议的绑定属性，要么可以得到 v 的一个解。考虑这样的情况： $Rsp_0=\left(r_0^{(0)},t_1^{(0)},e_1^{(0)}\right), Rsp_1=\left(r_1^{(1)},t_1^{(1)},e_1^{(1)}\right), Rsp_2=\left(r_1^{(2)},t_0^{(2)},e_0^{(2)}\right)$ 。可得

$$c_0 = Com(r_1^{(2)}, G(t_0^{(2)}, r_1^{(2)}) + e_0^{(2)}) = Com(r_1^{(1)}, v - F(r_1^{(1)} - G(t_1^{(1)}, r_1^{(1)}) - e_1^{(1)}))$$

$$c_1 = Com(t_0^{(2)}, e_0^{(2)}) = Com(r_0^{(0)} - t_1^{(0)}, F(r_0^{(0)}) - e_1^{(0)})$$

$$c_2 = Com(t_1^{(0)}, e_1^{(0)}) = Com(t_1^{(1)}, e_1^{(1)})$$

如果上面任意一个等式 Com 函数的两队参数是不同的，那么 Com 函数的计算绑定性被攻破。否则，可得出 $v = F\left(r_0^{(0)} + r_1^{(2)}\right)$ 。这意味着 v 的一个解 $r_0^{(0)} + r_1^{(2)}$ 被计算出来。

通过上面我们可以证明，对于一个不知道秘钥 v 的证明着来说，如果存在一个方案，对同一个 (c_0,c_1,c_2) 构造出 $ch=0, ch=1, ch=2$ 三种情况下都正确的副本，那么我们可以依照这个方案构造一个知识提取器，该知识提取器要么能够攻破 Com 函数的计算绑定性，要么能够在多项式时间范围内解决 MQ 问题。

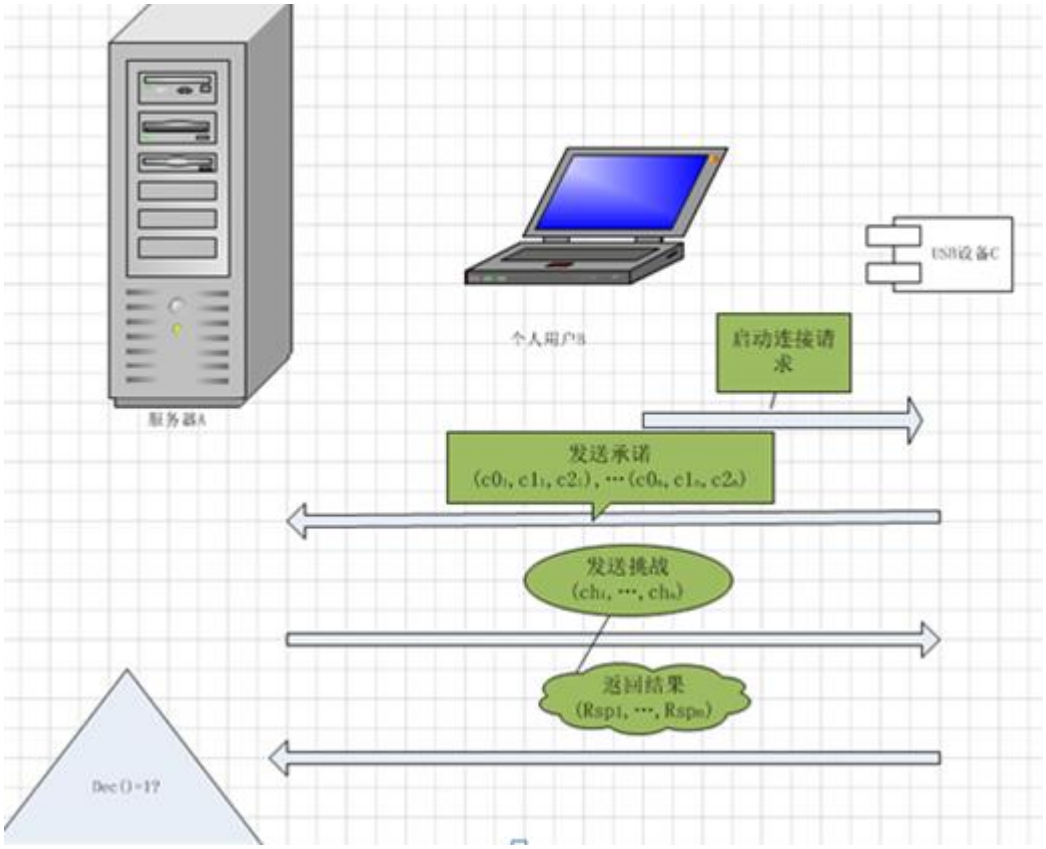
通过细致分析，我们可以知道，如果一个没有密钥 s 的用户想要伪装用户 P 与 V 进行交互。那么一次 3 步认证过程下来。P 被接受的概率是 2/3，多次重复这个交互过程，直到没有密钥 s 的用户只能以可忽略的概率欺骗用户 V 为止，完成协议。

3.2 FPGA 方案说明

本文中我们设计的 FPGA 上基于多变量公钥密码体制的身份认证系统是 Kitashinagawa 的 3 步认证方案的并行化设计，并将其具体实现在 FPGA 和计算机上。

3.2.1 参数说明

总体方案可以如图所示：



服务器 A：金融、政务等系统 OA 或交易服务器，用户登陆需要验证用户身份。他相当于身份认证系统中的验证者 V。

个人用户 B：个人用户计算机，服务器 A 的用户。服务 USB 设备 C 与服务器设备之前的通讯转发工作。

USB 设备 C：个人用户秘钥储存设备。负责承诺和挑战结果的计算，作为个人用户的身份证明。在身份认证系统中相当于证明者 P。

安全需求分析：

1.必须拥有 USB 设备才能完成人证过程。

2.USB 设备无法复制，既秘钥无法被获取。

密码方案需求：

1.个人用户 B 和服务器 A 之间的通讯是在不安全的网络中进行的，顾密码学身份认证方案应具有抗被动攻击能力。

2.由于个人用户或者服务器 A 都有可能被假冒，顾要求密码学身份认证方案应具有抗主动攻击能力

身份认证方案：

基于 MQ 问题的公钥认证方案。

并行三步并行构造认证方案。

设计参数：

MQ 函数： $MQ(160,80,F_2)$,

并行轮数：52 轮，达到 $2^{(-30)}$ 安全级别

承诺方案：基于 SHA1 算法的承诺方法
随机数： 由振荡器构成真随机数生成器，并由 LSFR 进行消偏处理。

3.2.2 并行安全性

单次（P，V）协议的执行，模仿者能够以 2/3 的概率欺骗验证者 V。只有通过单次协议的多次重复执行，才能将模仿者成功欺骗 V 概率降低。身份认证协议的多次重复执行，一般来说有顺序构造和并行构造两种方案。由零知识协议的顺序构造定理，我们可以知道顺序重复多次（P，V）协议所产生的新协议仍然是统计零知识的。而且，很显然顺序重复能够以最佳的速度降低知识错误。Bellare 和 goldreich 提出了通用的降低知识错误的顺序重复理论，当重复论述 $N = w(log\lambda)$ 时，身份认证系统就是在抗主动攻击的。虽然顺序重复多次执行 3 步协议构造简单、有效、安全，但是在密码学方案中可扩展性却不强。而且在实际使用中通讯次数太多也是一个问题。那么为了实际在 FPGA 上使用，并且能够将其扩展为数字签名方案和技术身份的数字签名方案。我们要考虑抗主公攻击的并行构造。

如果让底层 MQ 函数具有很强压缩性，特别的 F_q^n 到 F_q^m 的映射满足 n=m+k.并且 $k = w(log\lambda)$。在这养的条件下，3 步认证协议的并行构造（$Setup, Gen, P_N^{(p)}, V_N^{(p)}$）尽管私钥长度和通讯量相对于之前加倍增加了，但是它是抗主动攻击的。

这里选择 $MQ(160,80,F_2)$,作为困难问题的实例，并且其可证明是抗主动攻击的。因为 $MQ(160,80,F_2)$ 将 160bits 的密钥压缩到 80bits 的公钥，具有很强的压缩性，且对于同一个 v，仅存在一个 x 是的 $F(x)=v$ 的概率是可忽略的。同时我们通过简单计算可以知道，如果 $MQ(n,m,F_q)$ 是原像稳固的，那么 $MQ(n+1,m,F_q)$ 是二像稳固的。又因为强压缩性且是二像稳固的 MQ 函数我们可以看成是一个无爪函数。单次 (P,V) 协议具有知识性，即具有证据不可区分性，根据证据不可区分性的任意构造封闭性质可知并行执行 (P,V) 多次同样是证据不可区分的。有因为底层的困难问题实例是无爪的，因此我们可以得到并行执行 (P,V) 是证据隐藏的。

3.3 FPGA 实现

3.3.1 振荡器真随机数生成模块。

振荡器真随机数生成模块由控制模块、异或模块、振荡器模块、无反馈线性位移寄存器模块组成。通过振荡器模块产生真随机数源，然后利用线性反馈模块进行消偏。产生真随机数。

3.3.1.1 控制模块

控制内部随机数各个模块工作。利用 160 位位移寄存器保存随机数内容，每完成 160 位随机数，通过 state 信号通知外部模块获取随机数。

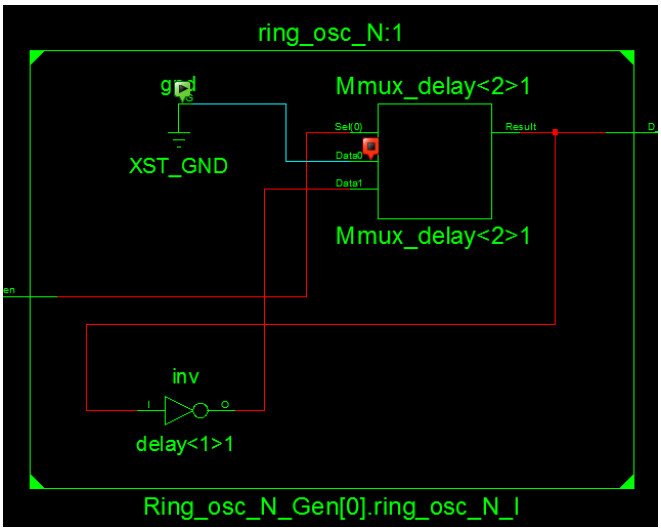
3.3.1.2 振荡器模块

本模块能产生不可以被预测的真随机数的根本原因在于本模块。其利用重复叠加反相器造成器件采样亚稳态源，它利用一个较慢的时钟去采集振荡器的输出获得真随机源。

算法：振荡器产生随机数

```
module rng_src
    # (
        parameter    WIDTH = 100 //震荡环数目
    )
    (
        output       d_out, //输出
        input        en,//使能信号
        input        clk//clk
    );

    wire [WIDTH-1:0] ring_o;

    genvar I;
    generate
        for( I=0 ; I<WIDTH ; I=I+1 ) begin : Ring_osc_N_Gen
            ring_osc_N ring_osc_N_I(     //生成 WIDTH 个震荡环，每个正当换的输出是 ring_o[I]
                .D_out   ( ring_o[I] ),
                .en      ( en )     //这些正当环的使能信号都与总的使能信号连接
            );
        end
    endgenerate

    wire     link_in_b;
    wire     link_in;

    xor_N # ( .N(WIDTH) ) //异或模块，参数代入。
    sum_up(
        .d_out   ( link_in_b ),
        .d_in    ( ring_o )   //输入时震荡换的输入
    );

    rmv_bias rmv_bias_inst(       //这个应该是随机数偏移模块
        .dat_o   ( d_out ),//这个是最后输出的随机数
        .dat_in  ( link_in_b ),//输入时异或模块的输入
        .clk ( clk )    );
```

```
endmodule
```

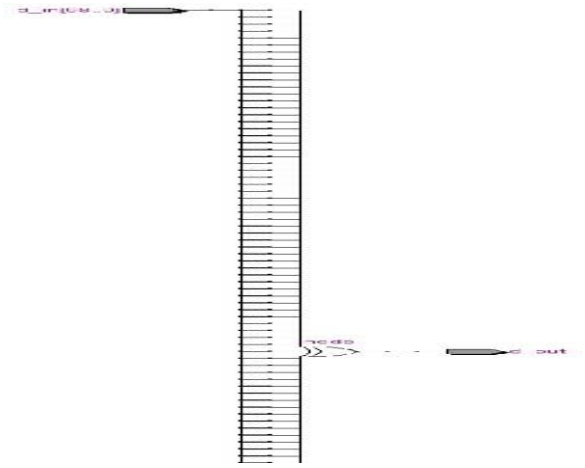


3.3.1.3 异或模块

异或模块将 100 个振荡器的输出结果进行异或，100 个随机源进行异或后获得更好的随机效果。本模块使用完全二叉树算法完成。

算法：异或模块

```
module xor_N
    # (
        parameter     N = 100
    )
    (
        output      reg       d_out,
        input       [N-1:0] d_in
    );

    always @* begin : complete_bin_tree //使用完全二叉树的方法
        reg [2*N-1:1] node;
        integer i;
        d_out <= node[i];
        node[2*N-1:N] <= d_in[N-1:0];
        for( i=1 ; i<N ; i=i+1 ) begin
            node[i] <= node[2*i] ^ node[2*i+1];
        end
    end

endmodule
```



3.3.1.4 无反馈线性位移寄存器

该模块将前面产生的随机源进行消偏，利用振荡器产生随机数容易导致随机数偏移现象严重，产生的真随机数可能具有一定的周期或模式。利用该无反馈线性位移寄存器对真随机源进行数学消偏，让随机源看上去更随机化。

算法：LSFR 消偏模块

```
module rmv_bias(
        output   dat_o,   //真随机数输出接口
        input    dat_in,
        input    clk
    );

    reg  [10:0]   tmp;

    initial begin
                tmp <= 11'b0;
    end
    always @(posedge clk) begin : reg_link
        integer i;
            for( i=1 ; i<11 ; i=i+1 )
                tmp[i] <= tmp[i-1];//然后一位一位移动
            tmp[0] <= dat_in;
      end

    assign dat_o = tmp[0] ^ tmp[3] ^ tmp[4] ^ tmp[6] ^ tmp[7] ^ tmp[8] ^ tmp[10]; //无反馈线性位移
endmodule
```

3.3.2 MQ 函数$\mathbf{F(x)}$、$\mathbf{G(x,y)}$向量计算等模块

该模块由保存系统参数 para 的 rom、F 计算模块、G 计算模块。采用 80 位计算结构，

利用 F、G 计算中参数的复用性，快速实现计算。

3.3.2.1 ROM 设计

$$\sum_{ij} a_{1ij}x_i x_j + \sum_i b_{1i}x_i = y_1$$

对于 MQ 方程，.... ，我们需要保存的在 ROM 中的系统参

$$\sum_{ij} a_{mij}x_i x_j + \sum_i b_{mi}x_i = y_m$$

数 aij,bj.对本方案 MQ(160,80,F2)。一共有 80 条方程。取其中第一条分析 ： $\sum_{ij} a_{1ij}x_i x_j + \sum_i b_{1i}x_i = y_1$

由于其对于 $x_j x_i$ 具有对成性，顾对不同的 ij 只需要保存一次 a_{ij}，将 b_i 看成是 x_i 与 $x_0(x_0=1)$ 的系数。则整个方程可以表示为 $\sum_0^n a_{ij}x_i x_j = y$（原本不存在 x0,如今加入 x0=1,让公式进行统一化）。则对一条方程的保存数据是:

$$\begin{pmatrix} a_{00} \\ a_{10}\ a_{11} \\ a_{20}\ a_{21}\ a_{22} \\ \cdots\cdots\cdots\cdots\cdots \\ a_{n0}\ \cdots\cdots\cdots\cdots a_{nn} \end{pmatrix}$$

整个 ROM 的结构是:

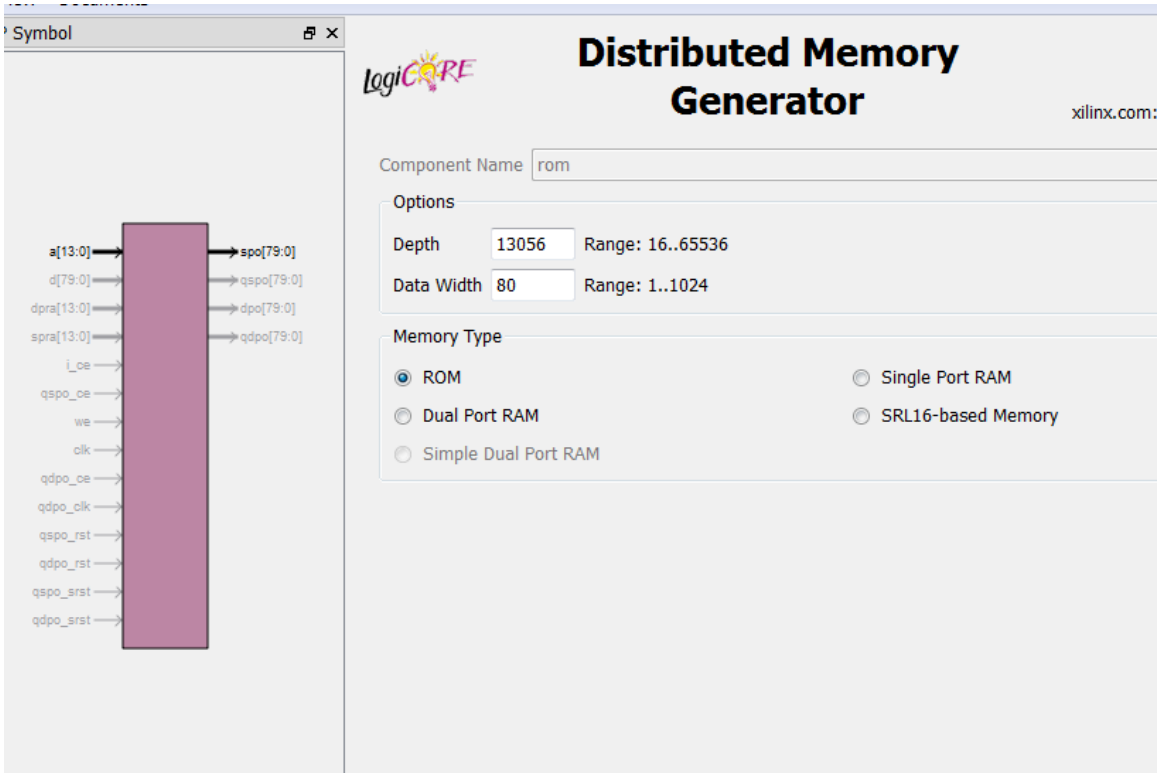
$$\begin{array}{cccc}
a_{00}^{(1)} & a_{00}^{(2)} & a_{00}^{(3)} & \cdots\cdots\ a_{00}^{(80)} \\
a_{10}^{(1)} & a_{10}^{(2)} & a_{10}^{(3)} & \cdots\cdots\ a_{10}^{(80)} \\
a_{11}^{(1)} & a_{11}^{(2)} & a_{11}^{(3)} & \cdots\cdots\ a_{11}^{(80)} \\
a_{20}^{(1)} & a_{20}^{(2)} & a_{20}^{(3)} & \cdots\cdots\ a_{20}^{(80)} \\
a_{21}^{(1)} & a_{21}^{(2)} & a_{21}^{(3)} & \cdots\cdots\ a_{21}^{(80)} \\
a_{22}^{(1)} & a_{22}^{(2)} & a_{22}^{(3)} & \cdots\cdots\ a_{22}^{(80)} \\
 & & \cdots\cdots & \\
a_{n0}^{(1)} & a_{n0}^{(2)} & a_{n0}^{(3)} & \cdots\cdots\ a_{n0}^{(80)} \\
 & & \cdots\cdots & \\
a_{nn}^{(1)} & a_{nn}^{(2)} & a_{nn}^{(3)} & \cdots\cdots\ a_{nn}^{(80)}
\end{array}$$

其中 $a_{ij}^{(w)}$ 表示第 w 条方程中， x_i 和 x_j 的系数，其存储量是

$$(n+1)*(n+2)/2*80=1043280 \quad \text{bits}$$

因方程 F、和 G 计算都存在并行性（$x_i x_j$ 可复用）。则 rom 设计为深度为 13041bits，宽度为 80（80 条方程并行）。

因 nexys3 没有单独足够大的 ROM，这里使用分布式设计 ROM。所以深度需达到 16 倍数。



3.3.2.2 F、G 计算模块。

F、G 模块利用 $x_i x_j$ 的复用性质，通过一次计算 80 条方程。提高计算速度。本模块计算一次需要 13041 时钟周期（该周期是 rom 时序周期，非全局时钟）

由于 GF(2)域中的计算可以看成是异或运算，所以无论加法与减法都可以用异或运算代替。

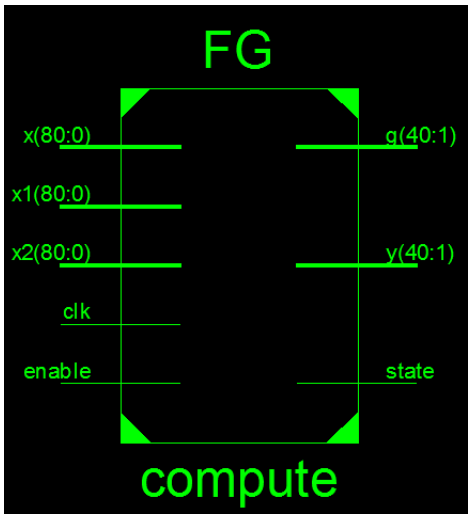
算法：FG 计算模块核心部分

```
begin
if(i==0&j==0)begin                    //
g1={80{(x1[i]&x2[j])^(x1[j]&x2[j])}}&douta; //这里计算 G
y1={80{x[i]&x[j]}}&douta;                   //这里计算 F
$display($time,"初始化存储器输出 douta%h addra%d i %d j%d",douta,addra,i,j);
end
else if(i>=j) begin
```

```
g1=g1 ^ ({80{(x1[i]&x2[j])^(x1[j]&x2[j])}}&douta); //这里计算 G
y1=y1 ^ ({80{x[i]&x[j]}}&douta); //这里计算 F。G 和 F 因为使用当相同的存储数据
                                  //所以这里能够同时进行计算，加快计算速度
$display($time,"存储器输出 douta%h addra%d i%d y%h,j%d",douta,addra,i,y,j);
end
end
```

3.3.2.3 FG 模块控制器

控制器控制 FG 模块内部读取 ROM 时序。同时计算结束后通过 state 信息通知外部控制器获得计算结果。模块图如下：



3.3.3 Rs232 协议数据收发模块

本模块由承诺发送模块、挑战接收模块、挑战发送模块构成。（其中承诺发送模块和挑战发送模块复用发送端口但不复用模块）统一采用 115200bps 作为通讯速率。负责与计算机进行通讯的工作。

3.3.3.1 承诺发送模块。

本模块以 160bit 的 SHA-1 散列数据位输入，调用内部 uart_tx 模块发送数据。发送单位为 byte.

3.3.3.2 挑战接收模块。

本模块通过监听 uart 口,按 rs232 协议监听来自计算机信息。以 byte 为接收单位，当收到 0、1、2 的 ASCII 码时记为挑战为 0、1、2.(这里因为上位机端使用 java，最小的处理单位是 byte，故造成了冗余)

算法：挑战接收模块

```
if(rx_state==1&count<=51)//&rx_data==8'b01100001
begin
rst_n=0;
$display($time,"完成一个字节的接收%h  计数器 count%d",rx_data,count);
```

```
if(rx_data==8'b00110000) begin    //计算机是按 byte 来发送挑战
challenge_reg[2:1]=00;            //这里将 byte 转换为两位的 bit
challenge_reg=challenge_reg << 2;
count=count+1;
end
if(rx_data==8'b00110001) begin
challenge_reg[2:1]=01;
challenge_reg=challenge_reg << 2;
count=count+1;
end
if(rx_data==8'b00110010) begin
challenge_reg[2:1]=10;
challenge_reg=challenge_reg << 2;
count=count+1;
end
end
if(count==53)   //到达最后一轮
begin
state_reg=1;   //设置完成状态，通知总控制器进行处理
$display($time,"挑战接收完成%h",challenge);
count=count+1;
end
end
```

3.3.3.3 应答发送模块

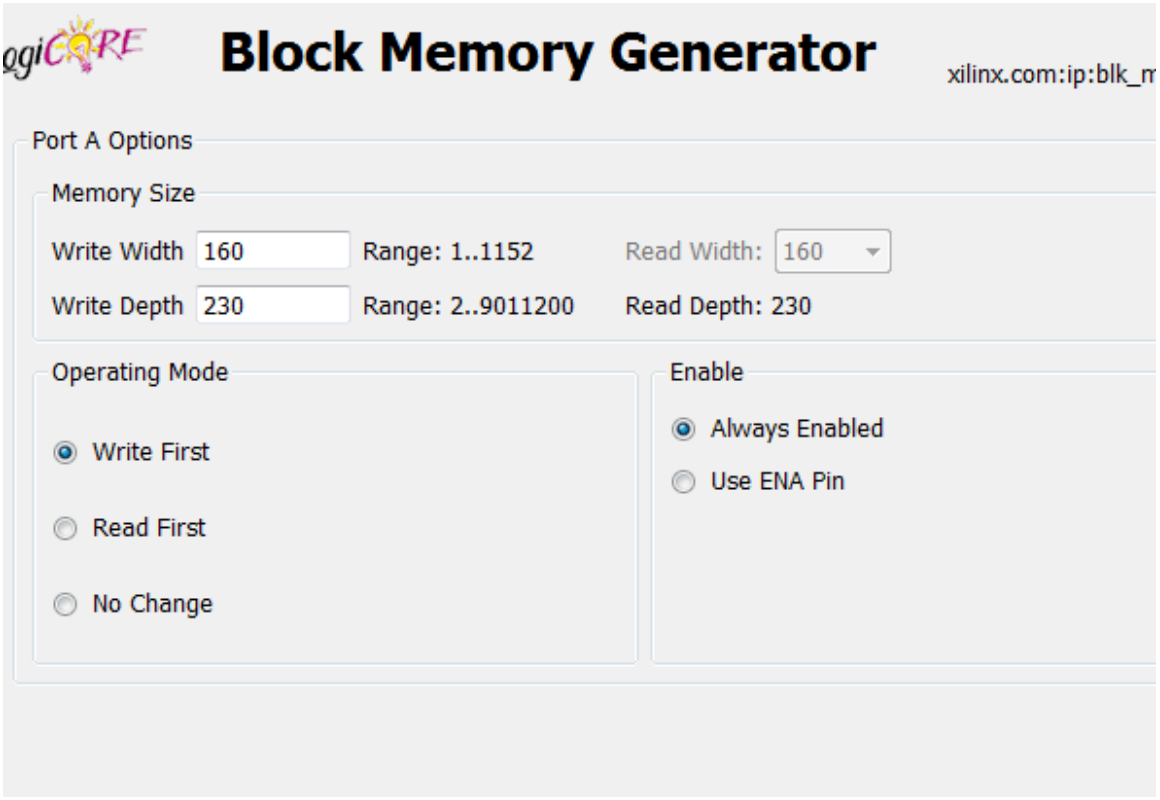
本模块对收到的挑战，读取 RAM 中的三元组信息对，并发送。

RAM 模块设计

普通 RAM 模块设计，每轮一次存储t_0, e_0, r_0, e_1，存储结构如下

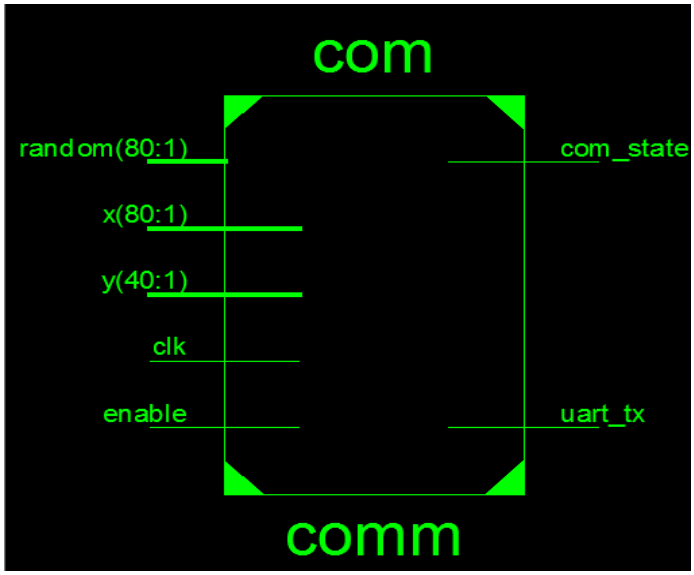
$$\begin{pmatrix} t_0^{(1)} \\ e_0^{(1)} \\ r_0^{(1)} \\ e_1^{(1)} \\ t_0^{(2)} \\ e_0^{(2)} \\ r_0^{(2)} \\ e_1^{(2)} \\ \cdots\cdots \\ t_0^{(52)} \\ e_0^{(52)} \\ r_0^{(52)} \\ e_1^{(52)} \end{pmatrix}$$

存储器宽度为 160bits，深度为 230。生成图如下：



3.3.4 Sha1 承诺模块

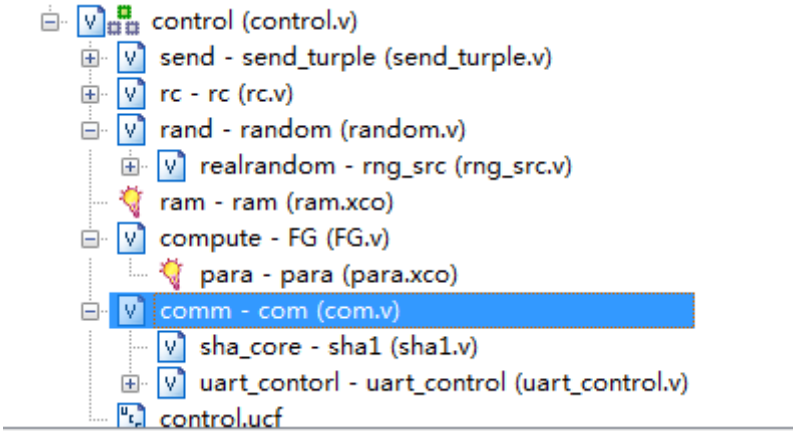
SHA1 算法：（secure hash algorithm）算法又称安全哈希算法。是由美国国家安全设计 NIST 发布一款密码学 hash 函数。目前 SHA1 是 SHA 函数中使用的最广泛的，被应用在许多协议和应用中。它能够产生 160-bit 消息散列，算法类似于 MD5 但是更加保守。本文中我们利用 SHA1 来作为承诺函数。这利用到了 SHA1 函数的抗碰撞的性质。本模块主要进行承诺计算。利用 sha1 IP 核进行改编。增加填充封装部分、随机数部分和时序控制部分，并与主控制器进行状态交互。



3.3.5 总控制器

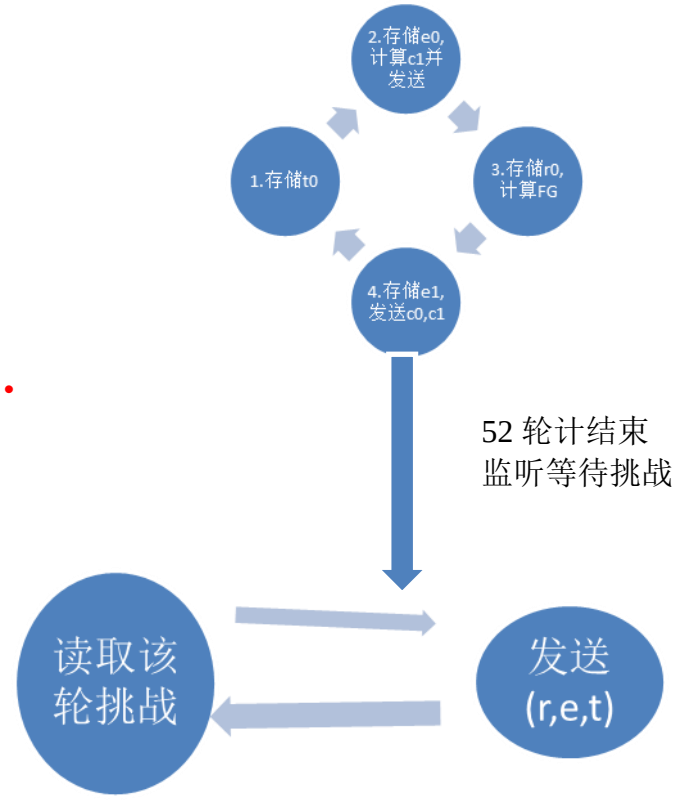
总控制器负责各个子模块之间状态变化沟通和控制。由于各个模块调用时间不同，总控制器仅实现部分流水化。总的模块结构如图

Control 为总控制器，其首先调用 random 模块获取随机数，然后将随机数模块生成的随机数分别输入 FG 计算模块，com 承诺模块。Com 承诺模块将承诺内容通过 uart 通



讯模块发送给计算机。完成承诺模块的发送后，控制器进入监听状态。当收到来来自 rc 挑战接收模块的中断信号后，控制器根据挑战值取 ram 中存储的计算值，并将结果通过 send 模块发送给计算机。

总状态变化图:



3.3.6 上位机验证模块

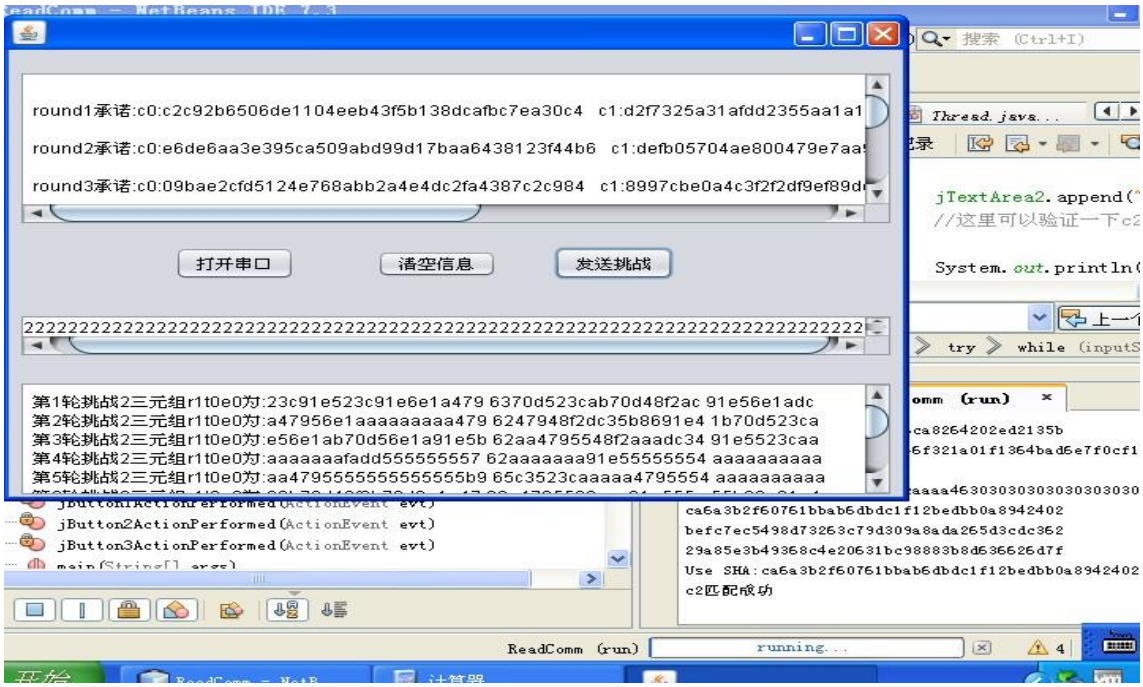
本模使用 java 编写，运行环境为 windows xp 系统调用 java 官方扩展包 javacomm20-win32 的接口函数进行 windows xp 系统与 FPGA 设备通过 com 口进行通讯。程序设计包括 rs232 发送和接收模块,sha1 承诺计算模块。程序运行后调用 rs232 监听模块对与 FPGA 设备连接的 USB 模拟 COM。选择与 FPGA 设备相同的波特率为进行通讯。当 FPGA 设备发送完 52 轮承诺值后，有控制者发送 52 个挑战给 FPGA 设备，并重新进入监听状态。FPGA 设备返回对应挑战的计算结果后，程序调用 sha1 算法计算承诺并判断是否与之前收到的承诺相等。如果相等，则验证通过，接收证明者身份。如果不相等，则拒绝证明者身份。

算法：上位机验证模块核心代码

```
for(x=0;x<52;x++) //共 52 轮
{
for(y=0;y<5;y++)
{
e[y]=readBuffer[x*25+4-y];
te[10+y]=readBuffer[x*25+4-y];    //对 FPGA 发送的 bits 冲排序进行转
}                                 //将其转换为原始的 e0、e1 的 java 语言能处理
的 byte 格式

for(y=0;y<10;y++)
{
r[y]=readBuffer[x*25+24-y];
t[y]=readBuffer[x*25+14-y];
te[y]=readBuffer[x*25+14-y];
                              }
for(y=15;y<25;y++)
{
  te[y]=0x30;
}
com=te.toString();
System.out.println(bytes2Hex(te));
String  cx=  c.substring(x,x+1);
TestEncrypt  te1=new  TestEncrypt();  //建立加密实例
System.out.println(comitment[3*x]); //输出之前收到的承诺
System.out.println(comitment[3*x+1]);
System.out.println(comitment[3*x+2]);
if(cx.equals("0"))  //目前测试中，r0  t0  e0  =1      r1  中间有 1 两边
0  t1  为 0  e1 是计算的，未知
{
jTextArea2.append("第"+(x+1)+"轮挑战"+cx+"三元组 r0t1e1
为:"+bytes2Hex(r)+"  "+bytes2Hex(t)+"  "+bytes2Hex(e)+"\n");
                                  //这里可以验证一下 c2,用 t1  e1
```

```
System.out.println("Use  SHA:"+te1.Encrypt(te,"SHA-1"));

if(te1.Encrypt(te,"SHA-1").equals(comitment[3*x+2]))
{
  System.out.println("c1 匹配成功  ");
                                    }
                                }
    if(cx.equals("1"))  {
jTextArea2.append("第"+(x+1)+"轮挑战"+cx+"三元组 r1t1e1
为:"+bytes2Hex(r)+"  "+bytes2Hex(t)+"  "+bytes2Hex(e)+"\n");
                                    //这里可以验证一下 c2,用 t1  e1
  System.out.println("Use  SHA:"+te1.Encrypt(te,"SHA-1"));
  if(te1.Encrypt(te,"SHA-1").equals(comitment[3*x+2]))
                                    {
                                            System.out.println("c1 匹配成
功  ");
                                    }
                                }
  if(cx.equals("2"))
  {
    jTextArea2.append("第"+(x+1)+"轮挑战"+cx+"三元组 r1t0e0
为:"+bytes2Hex(r)+"  "+bytes2Hex(t)+"  "+bytes2Hex(e)+"\n");
                                    //这里可以验证一下 c1,用 t0  e0
  System.out.println("Use  SHA:"+te1.Encrypt(te,"SHA-1"));
        if(te1.Encrypt(te,"SHA-1").equals(comitment[3*x]))
                                    {
          System.out.println("c2 匹配成功  ");
                                    }
                                }
                    }
```

程 序 运 行 截 图 如 下 ：



3.4 本章小节

本节主要介绍 FPGA 上基于多变量公钥密码的身份认证系统的具体实现算法。首先，我们对 FPGA 上基于多变量公钥密码的身份认证系统算法进行了详细的分析；然后根据实际需求设计出一个具体的在 FPGA 上身份认证系统，并分析了其安全需求，提出的具体的设计方案；然后我们将整个 FPGA 上的证明者程序划分成几个独立的模块，各个模块之间通过控制器进行协调工作，设计并加速了 FPGA 上多变量公钥密码身份认证系统的计算方法、随机数生成模块等核心模块，提高了计算的速度；最后我们在计算机上实现了验证者程序，并让其与 FPGA 设备通过 com 口进行通讯，进行交互式验证，并验证了 FPGA 程序的正确性。

第四章　实验结果

本节我们将对 FPGA 上程序进行的运行效率进行分析和总结，并通过分析指出目前工作存在不足指出和并听出可以改进的部位和方向。

4.1 随机数测试

本节我们使用 diehard 对随机数生成器进程随机性测试。Diehard 是一款测试随机数生成器质量的统计学测试软件。 Diehard 能够进行如 birthday spacings、overlapping permutations 等随机性测试，并输出 p 值。如果测试中存在超过 6 个项目的 P-value 值大于 1，

那么随机性测试不通过。本文中我们将随机数模块单独部署在 nexys3 板上，然后用 rs232 协议将其生成的随机数传到计算机中。每次收集 4Mbytes 数据进行测试。测试结果如下：

test		P-value	
		50 links	100 links
		25M	6.25M
birthday spacings		1	0.975562
overlapping permutations		1	0.997361
ranks of 31x31 matrices		0.484772	0.321205
ranks of 32x32 matrices		0.432009	0.348947
monkwy tests OPSO ,OQSO ,DNA		0.989968	0.976231
count the 1's in a stream of bytes		1	1
count the 1's in specific bytes		0.98989	0.975312
parking lot test		1	1
minimum distance test		1	1
random spheres test		0.997468	0.964051
the sqeeze test		1	1
overlapping sums test		0.823546	0.856231
runs test	runs up	0.879702	0.672959
	runs down	0.917223	0.085335
the craps test		0.998691	0.085335

我们采用 50 个震荡环和 25MHZ 采样频率进行采样时，测试结果如第一列。当我们采用 100 个震荡换和 6.25HZ 采样频率是，测试结果如第二列。通过数据我们可以知道该随机数生成器是满足随机性条件的。并且有采样频率越低，随机性越强；震荡环数越多，随机性越强。

4.2 性能分析

经过综合实现，该身份认证方案在 nexys3 芯片上资源总使用情况如下：

资源名	使用量	可用数	百分比
寄存器	3714	18224	20%
LUT 逻辑块	5126	9112	56%
IO 接口	5	232	2%
RAMB16BWERs	2	32	6%
RAMB8BWERs	1	64	1%

其中寄存器资源主要用于触发器资源，一少部分用于逻辑门电路。LUT 逻辑模块 99%用于逻辑构造，且其中 84%使用了 6 个输出口。虽然系统参数的存储量相当大，但

是实际上 FPGA 资源仍然是足够的。但设计总体使用 LUT 资源过多。

一下为程序运行时间情况：

模块名字	周期	备注
FG 模块	每 13042	Rom 的读取周期
随机数模块	每 400	输出 160bits
SHA1 模块	每 118	完成一次承诺计算
控制器	每 1	转换一次模块
发送模块	每 173820	发送 160bits
挑战接收		速率同发送

综合分析上列数字，总程序的各个子模块的逻辑单元使用情况和资源使用都相对较少，适合在小资源的平台上进行实现。但同时仍然可以发现，部分模块运行的时钟周期数较大。即便如此，在 100MHZ 主时钟频率的情况下，整个认证过程是十分快速的。

4.3 改进方向

通过前面的分析，我们大致可以看到目前方案仍然存在以下问题：FG 计算模块时钟周期数太多、rs232 通讯模块速度过慢、系统参数过大导致存储量过大、随机数生成器随机性不够。

随机数生成器随机性不够的原因是多方面的，震荡环个数、采样频率、FPGA 实验板性能、消偏处理函数等都对其有影响。震荡环个数增加和采样频率降低都有助于随机性的增强。但是这会带来吞吐量的降低和能耗的增加。在试验中如何平衡这几者的关系达到良好的随机性还需要进一步测试。

FG 模块时钟周期过长的原因是 MQ 认证方案本身系统参数太长的问题，本方案需要存储的公钥长度 130410bytes 长度，每次读取 80bits 进行计算，需要 13041 次。一种替代方案可以由有 80 个 LSFR 或者 NLSFR 来代替 rom 作为存储公钥的存储器。这可以大大降低时钟周期的长度，因为原采用 rom 存储公钥的方案在 FPGA 实现时其使用的是 sram，速度虽然快，但是在 100MHZ 的时钟下，无法到达 1 个周期读取 80bits。需要独立使用一个更慢的读取时钟来控制 rom,这造成了最终程序运行时间过长。若采用 LSFR 或者 NLSFR 代替 rom 进行公钥存储，不但可以极大的提高运行速度，且能够降低硬件资源和功耗，使其更适于低成本的硬件实现。下面我们给出一种减小存储资源的方案，该方案利用位移存储里将 1043280bits 的系统参数压缩到 160bits 的随机源。然后由一个 160bits 的线性位移存储器进行处理，每次出书 80bits 的系统参数。同时该方案由于线性位移存储器速度快于 ram 的速度，所以其能提升 3 倍的计算速度。方案代码如下。

算法：FG 计算模块核心部分

```
if(state_reg==0)
begin
   if(i==0&j==0)begin                //初始化
   g1={80{(x1[i]&x2[j])^(x1[j]&x2[j])}}&param[80:1]; //这里计算 G
   y1={80{x[i]&x[j]}}&param[80:1];                   //这里计算 F
    end
```

```
else if(i>=j) begin
g1=g1 ^ ({80{(x1[i]&x2[j])^(x1[j]&x2[j])}}&para); //这里计算 G
y1=y1 ^ ({80{x[i]&x[j]}}&para); //这里计算 F
end
 para={param[115:75]^param[160:120]^param[40:1],param[125:85]^param[154:114]};
param={para[80:1],param[160:80]};//对位移寄存器 param 进行变化，产生下一组参数
 if(i>j) //等于的时候不会执行这个操作。
 begin
  j=j+1;
 end
 else begin
    if(i==160)
      begin
          state_reg=1;
      end
     i=i+1;
     j=0;
end
```

Rs232 通讯模块速度过慢的主要原因在于其本身协议的限制，本方案采用波特率为 115200bps 的方案，这已经是 RS232 通讯中非常高的速度。因为 com 通讯口本身硬件限制，若再提高速度，将会造成通讯错误或硬件损坏。故要提升该模块速度可以考虑使用其他通讯协议，如 USB 协议等。这样可以极大的提高交互系统通讯速率。

4.4 本章小结

本章主要对真随机数生成器随机性、设备使用资源和运行周期进行简单分析和总结，提出了取代大存储量方案的和提高 FG 模块计算速度的另一可行方案，并分析了方案实现上存在的一些性能瓶颈，并指出可行的改进方向。

第五章　方案的扩展

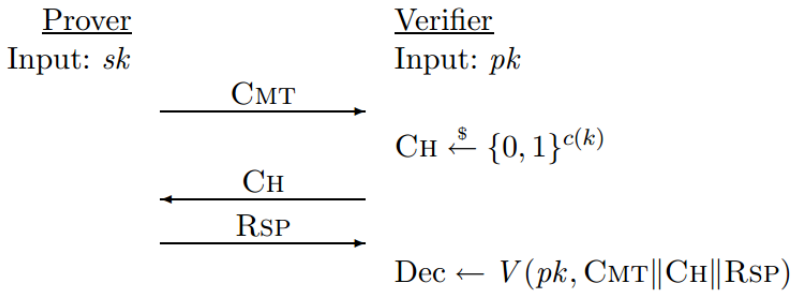
5.1 数字签名方案

FS 转换方法能够将身份认证方案变成一个非常有效的签名方案因此非常流行。本节中我们讲介绍如何使用 FS 转换将基于 MQ 的身份认证方案变成数字签名方案。并保证

其在随机预言模型下的安全性。下面我们先介绍一些背景知识并讨论一些已知结果，然后再转移到我们的结果上。

定义 5.1 （标准 ID 方案）：FS 转换通常应用在标准的三步认证方案中，这里我们称之为标准 ID 方案。一个标准的 ID 方案可以如下表示：$ID=(Setup,Gen,P,V)$ 。Setup 是系统初始化算法;Gen 是秘钥生成算法，返回公私钥对(pk,sk);P 是验证者算法，其输入为用户密钥 sk 和带前民；V 是一个决定性算法，以公钥 pk 为其输入，并计算出一个通讯脚本返回决定性的布尔决定。

标准的身份认证方案如下图：



由图可知，本文上两章讨论的基于 MQ 的身份认证系统是一个标准的身份认证方案。

定义 5.2 （数字签名方案）：数字签名方案可看成 $DS(Setup,Gen,S,V)$ 。Setup 是系统初始化算法；Gen 是秘钥生成算法返回公私钥对(pk,sk);S 是签名算法，输入 sk 和待签名消息 M，输出一个 M 的签名；V 是验证算法，其输入为 pk，消息 M 和签名 σ ，输出为一个布尔决定值，若为 1 则接受签名否则拒绝签名。

签名方案的安全目标：数字签名的安全目标是要让合法的签名者能够产生对某个消息的签名，不合法的用户无法对有效的消息产生签名，即签名的不可伪造性。这里我们介绍随机预言模型下不可伪造性。这要求敌手即使能够对签名者进行选择消息攻击并且能够访问随机 hash 函数的情况下，要对一个新的消息产生有效的签名在计算上是不可行的。

定义 5.3 （FS 转换）：签名者有用 ID 方案的公钥和私钥。签名一个消息 M 的时候，签名者像证明者一样计算出 Cmt 的值(挑战值)，并且将 $Cmt\|M$ 通过一个公共 hash 函数计算 hash 值，来获得一个挑战 $Ch=H(Cmt\|M)$,然后再像证明者一样计算出返回结果 Rsp ，然后将 M 的签名设置为 $Cmt\|Rsp$ Cmt||Rsp。为了验证 $Cmt\|Rsp$ 是否是 M 的签名，首先计算 $Ch=H(Cmt\|M)$,然后利用 ID 方案的验证函数 V，计算 $V(pk,Cmt\|Ch\|Rsp)$ 是否等于 1。

广义 FS 转换是由 Bellare 提出的 FS 转换的一种变形方案。Shamir 等人提出的标

准的 FS 转换是种子长度 s(k)为 0 的特殊实例。

定义 5.4 （广义 FS 转换）：令 $ID=(Setup,Gen,P,V)$ 是一个标准的身份认证方案，广义 FS 转换是对 FS 转换的一种修改。在普通 FS 转换中我们计算挑战值得时候使用 $Ch=H(Cmt\|\|M)$ 而在广义 FS 转换中，我们首先计算出一个长度为 $s(k)$ 的随机数 R，并在计算挑战之时将他代入 $Ch=H(R\|\|Cmt\|\|M)$; 由此来生成一个数字签名方案。

注意到签名算法是随机化的，使用一个长度为 s(k)加上证明着随机带的长度的随机带。更进一步，选择随机种子是签名的一部分，它使验证可行。Bellare 等人已经证明将广义 FS 转换应用于标准的身份认证方案中，当且仅当如果身份认证方案是在被动攻击模式下多项式安全，那么 DS 是在随机预言模型中选择消息攻击下多项式安全的。即身份认证方案的抗被动攻击性是广义 FS 转换方案获得安全的数字签名方案的最小化条件。

将广义 FS 转换应用于基于 MQ 的身份认证系统中，可以得到安全的数字签名方案。考虑到最小化安全条件，我们并不需要底层的身份认证方案具有抗主动攻击性。这样我们仅仅构造在被动攻击模式下安全的基于 MQ 的身份认证方案，便可获得抗选择明文攻击的数字签名方案。这样的好处是可以大大减小公钥大小、通讯量、计算次数。下面我们给出安全的设计方案。

我们考虑（P,V）的并行构造。（$Setup,Gen,P_N^{(p)},V_N^{(p)}$） 。很明显，对于一个诚实的验证者，（$P,V$）的并行重复仍保持零知识性。因为如果模拟器 S 知道 CV 将要选择的挑战 Ch，那么 S 总能输出成功的副本。并且，Pass 和 Venkitasubramaniam 指出在公开硬币知识论证中，并行重复在常数轮内减小知识错误。因此，身份认证协议 $(Setup,Gen,P_N^{(p)},V_N^{(p)})$ 在 $N=w(\log\lambda)$ 时是抗被动模仿攻击的。那么选用 $MQ(84,80,F_2)$ 作为底层 MQ 函数的身份认证方案，并重复执行 52 轮，可达到 $2^{(-30)}$ 安全级别，在被动攻击模式下是安全的。以下为其签名实现伪代码

算法:签名算法

```
输入: sk、M                  // sk 为密钥，M 为签署消息
输出: sig
R=random(s(k));              //获得长度为 s(k)的随机数
Rp=random(k);
Cmt=P(sk,Rp);                //计算承诺
Ch=H(R||Cmt||M);             //利用 hash 函数计算出挑战值
sig=P(sk,Cmt||Ch;Rp);        //计算出返回值
return sig;
```

算法:验证算法

```
输入: pk、M、sig              // pk 为公钥、M 为消息、sig 为待验证签名
```

输出: dec
R||Cmt||Rsp=sig; //解析 sig
Ch=H(R||Cmt||M); //计算出挑战值
dec=V(pk,Cmt||Ch||sig); //计算决定值，当 dec=1 接受签名，否则拒绝
return dec;

该方案的 FPGA 实现只需调用上章中身份认证方案的代码，只需改变 MQ 函数参数为 $MQ(84,80,F_2)$ 并使调用 SHA1 模块作为生成挑战的 hash 函数便可获得安全的数字签名方案。

5.2 基于 ID 的数字签名方案

基于 ID 的数字签名方案最先由 shamir 提出，其目的是为了简化密钥管理问题。在一个基于 ID 的数字签名方案中，一个用户的公钥就是他公开的身份（如身份证号码、电子邮箱等），再根据其公钥由信任中心来产生他的私钥。这样就可以在无公钥证书和复杂的密码学基础设施的情况下进行数字签名。

Bellare 等人提出了把一般签名转换为基于 ID 的签名的方法，但是其转换的前提条件是数字签名方案要满足可转换条件。如果存在一个陷门关系族与密钥生成函数 Gen 的输出同分布，那么我们说该方案是一个可转换的数字签名方案。将该条定义用于 MQ 上则可以理解为，如果要使用 cDS-2-IBS 转换将本方案生成的数字签名转换成基于 ID 的数字签名方案就需要能够找到一个能够解 MQ 方程的陷门，而这就强化了前提假设，而且需要 MQ 方程进行特殊构造。所以由标准的 cDS-2-IBS 生成基于 ID 的数字签名方案并不合适。但是这里我们可以很自然的利用证书来实现。

定义 5.5（基于证书的 IBS）: 给定一个标准的数字签名方案 $DS=(Setup,Gen,S,V)$，我们能够生成一个基于证书的 $IBS=(Mkg,Ukg,Sign,Vf)$ 。Mkg 主密钥生成算法，这里直接使用 Gen 来进行主密钥生成 (Msk,Mpk) 。Ukg 是用户密钥生成算法，他输出 $(sk,cert)$ ，sk 同样是由数字签名方案中的 Gen 算法生成的，cert 是用主私钥对身份 I 和用户公钥 pk 进行签名得到的证书和 pk 本身。用户签名是用自己的私钥 sk 进行签名并将 cert 一同发给对方。验证者首先主公钥 Mkg 验证 cert 的正确性，确保用户的公钥 pk 是真实的。如果用户的公钥 pk 是真实的，那么验证者在用该公钥验证

算法:Mkg 主密钥生成算法

输入: k //安全参数
输出: Msk、Mpk //系统主密钥和主公钥
R=random(); //获得长度为 s(k)的随机数
(Msk,Mpk)=Gen (k); //调用数字数字签名方案密钥生成算法生成主密钥

return Msk,Mpk

算法: Ukg 用户密钥生成算法

输入:I、k　　// I 为用户 id 信息
输出: sk,cert //sk 为用户密钥，cert 为证书
(sk,pk)=GEN(k);　　//调用 Gen 生成用户密钥信息
sig=S(Msk,pk||I);　　//调用 S 算法用主密钥对 pk 和 I 进行签名
cert=(pk,sig);　　//用户公钥的证书
return sk,cert

算法: Vf 签名检验算法

输入:M、signature、cert　　//M 为消息、signature 为签名、cert 为证书
输出: dec
(pk, sig)=cert;
If(sig!= V(Mpk,pk||I,sig))　　//用主公钥检验证书正确性
　Return 0;
If(V(pk, M,signature))　　//用验证正确后的用户公钥验证签名的正确性
　Return 1;　　//签名正确，返回 1
Else
　Return 0;　　//签名错误返回 0

5.3 本章小结

本章介绍了基于 MQ 问题的身份认证系统的方案的扩展，将广义 FS 转换其转换成数字签名方案，并使用了一种较自然的方法将其转换成基于身份的数字签名方案。

总结与展望

本文主要介绍了一种能够抗量子攻击的公钥密码身份认证方案，并在 FPGA 设备上将其实现。并对资源利用和性能进行分析。文章的最后还提出了该身份认证方案的两种扩展，尤其生成了数字签名方案和基于身份的数字签名方案。

本文的研究工作在 FPGA 实现上取得了一定得成果，但是仍然存在许多需要改进的地方。下面提出一些问题和改进的方向。

（1） 系统参数存储量太大，占用了大量 FPGA 资源。在不降低方案安全性的情况下降低系统参数的存储是一个优化的重点。

（2） 模块运算周期数过多，需要进一步减小。

（3） Uart 通讯模块速率太低，是目前方案在速度上的瓶颈。采用 USB 或其他速度更快的接口进行通讯，来提高通讯速率。

（4） 控制器未流水化，造成总吞吐量降低。进一步流水化各个模块，提高系统每个部件的使用率，从而提高吞吐速率和降低功耗。

（5） 相应的基于身份的数字签名方案仍然用到了证书。主要问题在于方案底层数学原理需从构造角度重新优化构造方案。

结 论

参考文献

1] Shamir A., ‘Identity based cryptosystems and signature schemes’, Cryptology -Crypto’84, LNCS 196, Springer-Verlag, pages 47–53.

[2] Bellare M, Goldreich O. On defining proofs of knowledge[C]//Advances in Cryptology—CRYPTO’92. Springer Berlin/Heidelberg, 1993: 390-420.

[3] Bellare M, Palacio A. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks[J]. Advances in Cryptology—CRYPTO 2002, 2002: 149-162.

[4] Bettale L, Faugère J C, Perret L. Hybrid approach for solving multivariate systems over finite fields[J]. Journal of Mathematical Cryptology, 2009, 3(3): 177-197.

[5] Feige U, Fiat A, Shamir A. Zero-knowledge proofs of identity[J]. Journal of cryptology, 1988, 1(2): 77-94.

[6] Feige U, Shamir A. Witness indistinguishable and witness hiding protocols[C]//Proceedings of the twenty-second annual ACM symposium on Theory of computing. ACM, 1990: 416-426.

[7] Fiat A, Shamir A. How to prove yourself: practical solutions to identification and signature problems[C]//Advances in Cryptology—Crypto’86. Springer Berlin/Heidelberg, 1987: 186-194.

[8] Goldreich O. Foundations of cryptography[J]. fragments of a book, 1995, 346.

[9] Lyubashevsky V. Lattice-based identification schemes secure under active attacks[J]. Public Key Cryptography–PKC 2008, 2008: 162-179.

[10] Sakumoto K, Shirai T, Hiwatari H. Public-key identification schemes based on multivariate quadratic polynomials[J]. Advances in Cryptology–CRYPTO 2011, 2011: 706-723.

[11] Berbain C, Gilbert H, Patarin J. QUAD: A practical stream cipher with provable security[J]. Advances in Cryptology-EUROCRYPT 2006, 2006: 109-128.

[12] Arditti D, Berbain C, Billet O, et al. Compact FPGA implementations of

QUAD[C]//Proceedings of the 2nd ACM symposium on Information, computer and communications security. ACM, 2007: 347-349.
[13] 张润捷. 一种基于 FPGA 实现的真随机数发生器[J]. 中国集成电路, 2009, 17(11): 52-55.
[14] 霍文捷, 刘政林, 陈毅成, 等. 一种基于 FPGA 的真随机数生成器的设计[J]. 华中科技大学学报: 自然科学版, 2009, 37(1): 73-76.
[16]Ding J., Schmidt D.. Multivariate Public Key Cryptosystems[M]. Berlin:Springer, 2009.