**2024年全国大学生信息安全竞赛**

**作品报告**

**作品名称： 零知识友好型哈希函数的算术电路实现与优化**

**电子邮箱： 3438772254@qq.com**

**提交日期：2024年6月5日**

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者可以在此基础上增加内容或对文档结构进行微调。

5. 为保证网评的公平、公正，作品报告中应避免出现作者所在学校、院系和指导教师等泄露身份的信息。

目录

[摘要 1](#_Toc11302)

[Abstract 3](#_Toc29983)

[第一章 作品概述 6](#_Toc16822)

[1.1背景分析 6](#_Toc5275)

[1.1.1 零知识证明广受关注 6](#_Toc9311)

[1.1.2 区块链技术迅速发展 8](#_Toc1131)

[1.1.3 当前业界哈希算法存在缺陷 8](#_Toc31788)

[1.2研究现状 9](#_Toc15208)

[1.2.1 零知识证明与zk-SNARK 9](#_Toc32681)

[1.2.2 circom 9](#_Toc3257)

[1.2.3 Rescue-Prime算法 10](#_Toc30974)

[1.2.4 circom实现的rescue哈希算法优势 11](#_Toc22272)

[1.2.5 与其他哈希算法对比 11](#_Toc31995)

[1.3作品概述 14](#_Toc4533)

[1.4前景分析 15](#_Toc11129)

[第二章 作品设计与实现 17](#_Toc21129)

[2.1系统概述 17](#_Toc19379)

[2.2系统需求分析 18](#_Toc29395)

[2.2.1 功能需求分析 18](#_Toc27788)

[2.2.2 安全需求分析 20](#_Toc32070)

[2.2.3 非功能需求分析 20](#_Toc12250)

[2.3技术背景 21](#_Toc3720)

[2.3.1 零知识证明发展与应用 21](#_Toc24877)

[2.3.2 circom语言基础 22](#_Toc26471)

[2.3.3 零知识友好哈希函数Rescue-prime 22](#_Toc15274)

[2.4预备知识 23](#_Toc9657)

[2.4.1 符号定义 23](#_Toc8184)

[2.4.2 零知识证明 24](#_Toc294)

[2.4.3 R1CS（一阶约束系统）算术电路介绍 28](#_Toc16973)

[2.4.4 零知识友好型哈希算法 29](#_Toc4980)

[2.5关键设计 30](#_Toc23065)

[2.5.1 电路层次结构设计 30](#_Toc12303)

[2.5.2 rescue-prime安全性分析 33](#_Toc26600)

[2.5.3 零知识证明电路的实现 35](#_Toc11486)

[第三章 作品测试与分析 43](#_Toc2925)

[3.1测试环境 43](#_Toc20127)

[3.2功能测试 44](#_Toc29400)

[3.3性能测试 50](#_Toc23916)

[第四章 创新性说明 56](#_Toc30180)

[4.1Rescue算法在零知识证明系统中的应用与优化 56](#_Toc18961)

[4.2circom语言中的Rescue-Prime实现 57](#_Toc16956)

[4.3高效且安全的R1CS电路设计 57](#_Toc21334)

[第五章 总结 59](#_Toc27690)

[参考文献 61](#_Toc21109)

插图目录

[图1 隐私泄露词条的搜索变化](#_Toc19721)[[2]](#_Toc19721) [6](#_Toc19721)

[图2 零知识证明词条的搜索变化](#_Toc9457)[[2]](#_Toc9457) [6](#_Toc9457)

[图3 零知识证明应用](#_Toc11195)[[3]](#_Toc11195) [7](#_Toc11195)

[图4 circom结构](#_Toc14343)[[5]](#_Toc14343) [10](#_Toc14343)

[图5 Algebraic Execution Time:AET 12](#_Toc26062)

[图6 系统模块 18](#_Toc25691)

[图7 功能需求 19](#_Toc19716)

[图8 zk-snark的大致流程 26](#_Toc4221)

[图9 算术电路示例图 27](#_Toc14336)

[图10 海绵结构的基本构造 30](#_Toc24750)

[图11 电路的垂直层设计图 31](#_Toc19137)

[图12 第 i 轮Rescue-XLIX置换过程示意图 32](#_Toc25485)

[图13 Rescue-Prime的海绵结构迭代过程示意图 32](#_Toc10346)

[图14 Rescue-prime Optimized流程改进 43](#_Toc29171)

[图15](#_Toc29252) [基于Groth 16协议的零知识证明过程 45](#_Toc29252)

[图16 编译完成后的电路信息 46](#_Toc17947)

[图17 可信设置，生成τ的幂次 47](#_Toc14341)

[图18 pot14\_003.ptau文件验证成功 48](#_Toc3590)

[图19 pot14\_final.ptau文件验证成功 49](#_Toc15699)

[图20 生成初始的zkey文件 50](#_Toc32029)

[图21 导出验证密钥 50](#_Toc14468)

[图22](#_Toc8473) [验证成功 51](#_Toc8473)

[图23](#_Toc28328) [circom文件编译信息 52](#_Toc28328)

[图24](#_Toc32288) [运行时间 53](#_Toc32288)

[图25](#_Toc20681) [非线性约束数量 54](#_Toc20681)

[图26](#_Toc13872) [证明密钥大小 55](#_Toc13872)

表格目录

[表1](#_Toc8225) [各zk-friendly hash函数的效率对比 12](#_Toc8225)

[表2](#_Toc21721) [符号定义 24](#_Toc21721)

[表3](#_Toc12735) [测试环境 42](#_Toc12735)

[表4](#_Toc43) [四种哈希算法的编译时间（单位：s） 49](#_Toc43)

[表5](#_Toc14175) [四种哈希算法的运行时间（单位：s） 50](#_Toc14175)

[表6](#_Toc9623) [四种哈希算法的非线性约束数量 51](#_Toc9623)

[表7](#_Toc11848) [四种哈希算法生成的证明密钥大小（单位：kB） 51](#_Toc11848)

# 摘要

随着零知识证明（Zero-Knowledge Proof，ZKP）在可信计算、区块链扩容和隐私计算等多个领域的广泛应用，哈希函数作为重要的密码学原语，在哈希原像零知识证明中具有广泛的应用场景。然而，由于ZKP应用需要将待证明问题转化为ZK电路（例如 r1cs）的形式，像 SHA-256 这样的标准哈希算法在原像零知识证明中的效率较低，无法满足现有ZK应用的需求。为此，人们提出了各种更适用于ZK计算的哈希算法，例如 MIMC、Poseidon 和 Rescue。在R1CS（SNARK）中，Rescue相对于MiMC有接近58%的效率优势，在AET（STARK）中，这一优势更是高达接近89%。尽管 Rescue 算法在效率方面表现突出，但目前工业界普遍使用的是 MIMC 和 Poseidon。因此，本作品将探讨如何将适用于零知识证明的Rescue-prime哈希函数转化为ZK电路。

本作品的主要创新点在于成功地使用circom语言实现了Rescue-prime哈希函数。由于Rescue-prime哈希函数的ZK电路实现比较困难，导致没有人在当前流行的区块链平台上进行过实现，故本作品将创新性地使用circom实现适用于零知识证明的Rescue-prime哈希函数。circom是一种专门用于编写和优化零知识证明电路，其简洁和高效的特性使其成为构建复杂密码学电路的理想工具。本作品克服了circom语言实现Rescue-prime算法的各种困难，包括circom稀少的库函数和Rescue-prime逆S盒步骤的大指数等等。本作品通过创新性地将许多不必要的电路实例化的方法采用function实现而非template，成功地克服了这些困难。这种方法不仅提高了电路的效率，也提高了其可读性。此外，本作品还采用了快速模幂的思想，并开设了二维数组，以克服circom语言不允许对信号多次赋值的问题。并且本作品在完成以上适用于STARK的初步设计后，学习针对rescue-prime的进一步优化，Rescue-Prime Optimized并初步采用其思想。

本作品的优势体现在采用circom而非其他零知识编程语言、实现rescue-prime哈希函数而非其他哈希函数。为阐述其优势，本作品对circom和rescue prime均作详细介绍，并将他们与同领域技术或者工具作对比。circom工具具备专注于电路设计、易于学习、集成优化策略（减少约束数量）、具备开源库四个优势。Rescue-Prime 算法的核心优势在于其为算术化导向的哈希函数，这意味着它特别适用于在密码学协议中进行评估，如多方计算或零知识证明系统。这种算法的设计允许它在算术表示（arithmetization）中有一个紧凑的描述，从而减少了相关协议的复杂性。Rescue哈希函数本身在效率方面即由于Mimc、Poseidon等零知识友好型哈希函数。而rescue-prime算法又对其做了在素数域上实现等改进。

本作品的具体工作是使用circom实现Rescue-prime哈希函数电路并对其做测试和优化。在本作品中依次对Rescue-prime哈希函数的S盒模块、MDS变换层、轮常量加、逆S盒四个模块进行了实现。实现中的每个模块都精心设计，以确保其在零知识证明应用中的性能和安全性。随后，本作品对实现的Rescue-prime哈希函数电路进行了性能评估和验证。并同普通哈希函数SHA-256、零知识友好哈希函数rescue、零知识友好函数Poseidon进行对比。实验结果表明，该电路在执行哈希计算时相较其他哈希函数具有更高的计算效率（SHA256五倍以上）、更少的约束数量（SHA256的50%以下），能够满足零知识证明场景中的需求。

总的来说，本作品弥补了circom这一重要零知识平台上缺失的关键一环。使用circom完成一种暂未被实现的优秀零知识哈希算法rescue-prime。通过本作品的工作，将为业界提供更高效的零知识哈希函数。并且随着零知识证明的广泛应用，本作品相信本作品的前景必将十分广阔。希望可以为读者提供使用circom实现哈希函数的一个思路，以应对更复杂的密码学需求。本作品期待着未来能够看到更多的应用和研究，以进一步推动这一领域的发展。本作品也期待着您的反馈和建议，以帮助本作品改进和完善本作品的工作。感谢您的阅读和支持。

**关键字：**零知识证明，哈希函数，circom，Rescue哈希算法

# Abstract

With the widespread application of Zero Knowledge Proof (ZKP) in various fields such as trusted computing, blockchain scalability, and privacy computing, hash functions, as important cryptographic primitives, have a wide range of application scenarios in hash like zero knowledge proof. However, due to the need for ZKP applications to transform the problem to be proven into the form of ZK circuits (such as r1cs), standard hash algorithms like SHA-256 have lower efficiency in original image zero knowledge proof and cannot meet the needs of existing ZK applications. For this reason, various hash algorithms that are more suitable for ZK computing have been proposed, such as MIMC, Poseidon, and Rescue. In R1CS (SNARK), Rescue has a nearly 58% efficiency advantage over MiMC, and in AET (STARK), this advantage is as high as nearly 89%. Although the Rescue algorithm performs outstandingly in terms of efficiency, MIMC and Poseidon are currently widely used in the industry. Therefore, this work will explore how to transform the Rescue prime hash function applicable to zero knowledge proof into a ZK circuit.

The main innovation of this work lies in the successful implementation of the Rescue prime hash function using circom language. Due to the difficulty in implementing the ZK circuit of the Rescue prime hash function, no one has implemented it on the current popular blockchain platform. Therefore, this work will innovatively use Circle to implement the Rescue prime hash function suitable for zero knowledge proof. circom is a specialized tool for writing and optimizing zero knowledge proof circuits, and its concise and efficient characteristics make it an ideal tool for building complex cryptographic circuits. We have overcome various difficulties in implementing the Rescue prime algorithm in circom language, including the scarcity of library functions in circom and the large exponent of the Rescue prime inverse S-box step. We have successfully overcome these difficulties by innovatively implementing many unnecessary circuit instantiation methods using functions instead of templates. This method not only improves the efficiency of the circuit, but also enhances its readability. In addition, we also adopted the idea of fast exponentiation and created a two-dimensional array to overcome the problem of circom language not allowing multiple assignments to signals. And this work learns and draws on the latest optimization of the rescue prime hash algorithm.

The advantages of this work are reflected in the use of Circle instead of other zero knowledge programming languages, and the implementation of the rescue prime hash function instead of other hash functions. To illustrate their advantages, this work provides a detailed introduction to both Circle and Rescue Prime, and compares them with technologies or tools in the same field. The circom tool has four advantages: focusing on circuit design, easy to learn, integrating optimization strategies (reducing the number of constraints), and having open source libraries. The core advantage of the Rescue Prime algorithm lies in its arithmetic oriented hash function, which means it is particularly suitable for evaluation in cryptographic protocols such as multi-party computation or zero knowledge proof systems. The design of this algorithm allows for a compact description in arithmetic representation, thereby reducing the complexity of related protocols. The Rescue hash function itself is efficient due to zero knowledge friendly hash functions such as Mimc and Poseidon. And the rescue prime algorithm has made improvements such as implementation in the prime field.

The specific work of this work is to use circom to implement the Rescue prime hash function circuit and test it. In this work, the S-box module, MDS transformation layer, round constant addition, and inverse S-box modules of the Rescue prime hash function are implemented sequentially. Each module in the implementation is carefully designed to ensure its performance and security in zero knowledge proof applications. Subsequently, this work evaluated and validated the performance of the implemented Rescue prime hash function circuit. Compare it with the ordinary hash function SHA-256, the zero knowledge friendly hash function rescue, and the zero knowledge friendly function Poseidon. The experimental results show that this circuit has higher computational efficiency (more than five times SHA256) and fewer constraints (less than 50% of SHA256) compared to other hash functions when performing hash calculations, and can meet the requirements of zero knowledge proof scenarios.

Overall, this work fills a crucial gap in the important zero knowledge platform of Circle. Use circom to implement an excellent zero knowledge hash algorithm called rescue prime, which has not yet been implemented. Through the work of this work, we will provide the industry with more efficient zero knowledge hash functions. And with the widespread application of zero knowledge proof, we believe that the prospects of this work will be very broad. I hope to provide readers with an idea of using circom to implement hash functions, in order to meet more complex cryptographic requirements. We look forward to seeing more applications and research in the future to further promote the development of this field. We also look forward to your feedback and suggestions to help us improve and perfect our work. Thank you for your reading and support.

**Keywords**: Zero-Knowledge Proof, Hash Function, circom,Rescue Hash Algorithm

# 第一章 作品概述

## 1.1背景分析

### 1.1.1 零知识证明广受关注

隐私问题一直是社会关注的焦点，尤其是在数字化和网络化的今天，这个问题显得更加重要。随着网民素质的提高和隐私意识的提升，人们对于保护个人隐私的需求越来越强烈。此外，随着互联网技术的普及和各种互联网行动的广泛开展，人们的线上活动越来越多，这也使得隐私保护的需求更加迫切。

Google Trends是一个能够反映人们搜索行为的工具，它可以在一定程度上反映出人们对于某一主题的关注程度。Google Trends的数据从侧面反映出人们对于隐私保护的重视。

如图1所示，可以看到在过去的一段时间内，"隐私"这一关键词的搜索量一直保持在较高的水平，这说明人们对于隐私问题的关注并未减少，反而在不断增加。这也说明，随着人们对于隐私意识的提高，以及互联网行动的普及，隐私保护的需求将会更加强烈。

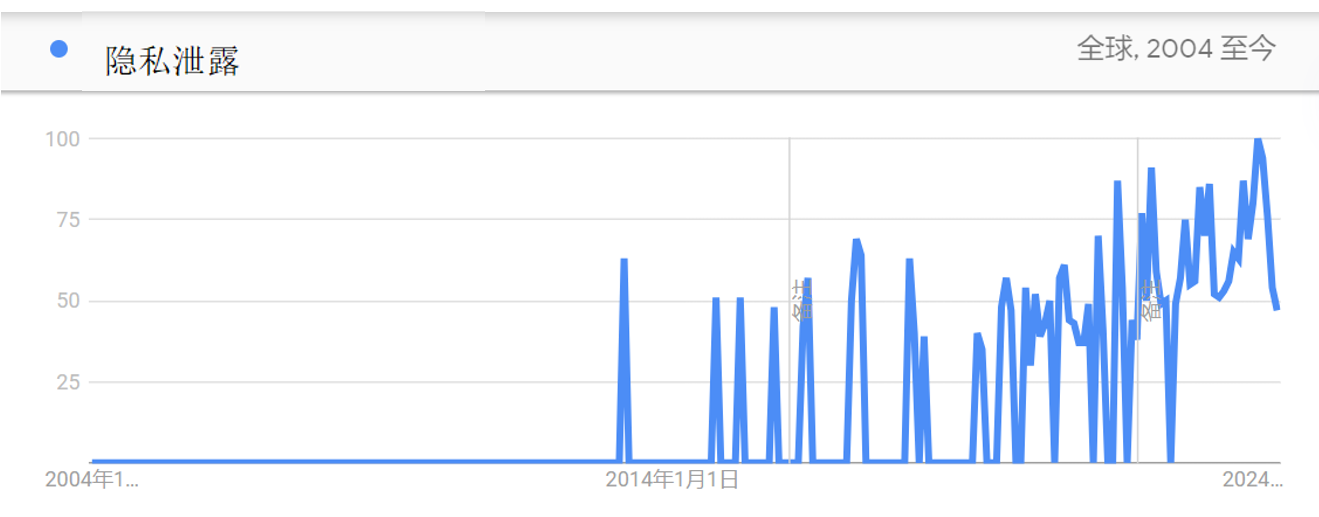


图1 隐私泄露词条的搜索变化[2]

零知识证明（ZKP）技术，作为密码学领域的一项创新，近年来在保护隐私的同时验证信息的正确性方面显示出巨大潜力，同样也得到广泛关注，如图2。

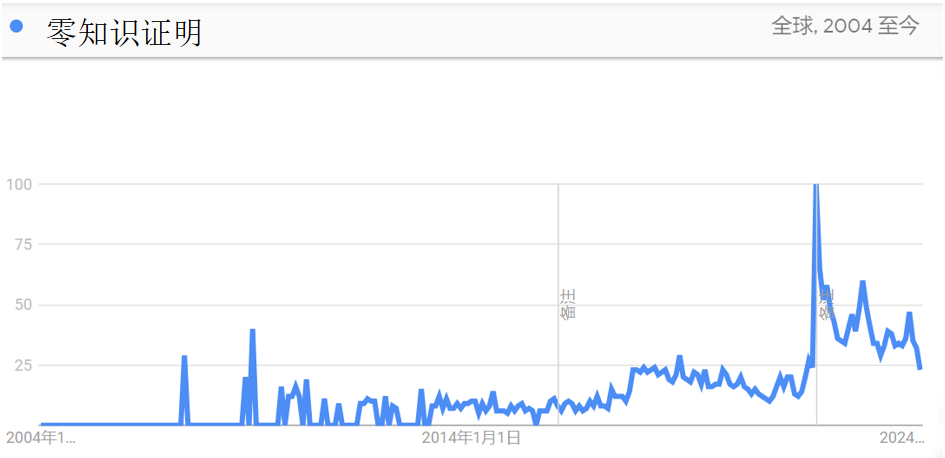


图2 零知识证明词条的搜索变化[2]

零知识证明之所以能够保护隐私，是因为它具备以下特点：

不透露额外信息：零知识证明的核心特性是能够在不泄露任何敏感信息的情况下进行数据验证，这大大提高了数据的安全性和隐私性。

匿名认证：在身份验证系统中，零知识证明允许用户证明他们属于某个群体或拥有某项权利，而无需透露具体身份或其他个人信息。

保护敏感数据：在医疗、金融等领域，零知识证明可以用来验证个人的资格或交易的合法性，同时保护个人的医疗记录和财务信息不被第三方获取。

去中心化应用：零知识证明还可以用于构建去中心化应用程序（DApps）和智能合约，使得这些应用程序可以在不泄露任何敏感信息的情况下进行交互和验证。

交互式证明系统：零知识证明通常涉及复杂的密码学概念，如承诺方案和交互式证明系统。在这些系统中，证明者和验证者通过一系列交互来达成共识，证明者证明了某个陈述的真实性，而验证者获得了足够的信心，但两者都没有泄露任何额外的信息。这种技术的兴起，与信息技术的飞速发展和社会对隐私保护需求的增加密切相关。在区块链技术的推动下，ZKP已经开始在金融、供应链管理、身份验证等多个领域发挥作用。



图3 零知识证明应用[3]

### 1.1.2 区块链技术迅速发展

随着数字经济的蓬勃发展，区块链技术作为其核心支撑，已经渗透到金融、供应链、身份验证等多个领域。在这个过程中，隐私保护和数据安全成为了人们最关注的问题之一。零知识证明技术，因其能够在不泄露任何额外信息的前提下验证信息的真实性，成为了解决这一问题的关键技术。

在区块链领域，零知识证明的应用主要集中在提升交易的隐私性和安全性。例如，去中心化金融（DeFi）项目通过采用ZKP，可以在不透露用户身份和交易细节的情况下，完成交易验证。这不仅保护了用户的隐私，也增强了整个系统的安全性。

此外，随着区块链技术的不断进步，主流区块链平台也开始集成零知识证明相关的技术，如zk-SNARKs和zkRoll-ups，以提高网络的扩展性和性能。这些技术使得可以在保持数据隐私的同时，处理更多的交易，从而解决了区块链可扩展性的挑战。

总之，区块链对于零知识证明的需求源于对隐私保护和数据安全的迫切需要。随着技术的不断发展，零知识证明有望在未来的区块链应用中发挥更加重要的作用，为创建一个更安全、更私密、更高效的数字经济生态系统提供支持。

### 1.1.3 当前业界哈希算法存在缺陷

在密码学中，哈希算法是一种关键的工具，它可以将任意长度的输入（也称为消息）转化为固定长度的输出。其中，SHA256是一种被广泛使用的标准哈希算法，它的输出长度为256位。然而，在原像零知识证明（zk-SNARKs）中，SHA256的效率较低，无法满足现有零知识证明应用的需求。

零知识证明是一种密码学协议，允许一方（证明者）向另一方（验证者）证明一个陈述是真实的，而无需提供任何有关该陈述的其他信息。在这种情况下，需要一种更高效的哈希算法。

为了解决这个问题，研究人员提出了各种更适用于零知识计算的哈希算法，如MIMC、Poseidon和Rescue。这些算法都是为了在零知识证明环境中提供更高的效率和安全性。

MIMC是一种低度哈希函数，它的主要优点是在有限字段上的计算效率高。Poseidon是另一种新型哈希函数，它利用了SPN结构和新的哈希策略，以提高在零知识证明中的性能。

Rescue算法是最近提出的一种哈希算法，它在效率方面表现突出。Rescue，通过在每轮中交替使用两种不同的函数来提高安全性，同时保持了较高的效率。

然而，尽管Rescue在理论上具有较高的效率，但目前工业界普遍使用的还是MIMC和Poseidon3。这可能是因为这两种算法已经被广泛研究和测试，而Rescue算法还相对较新，需要进一步的研究和验证。总的来说，选择哪种哈希算法取决于具体的应用需求和环境。在未来，期待看到更多高效且安全的哈希算法被提出和应用在零知识证明中。

## 1.2研究现状

### 1.2.1 零知识证明与zk-SNARK

零知识证明是一种密码学协议，允许证明者向验证者证明某个陈述的真实性，同时不泄露任何有关该陈述的实际信息[1]。这意味着验证者可以确认陈述的真实性，但不知道秘密向量witness。零知识证明有广泛的应用，包括密码学、区块链、身份验证等领域。在密码学中，它可以用于验证密码学协议的安全性，而在区块链中，它可以用于实现隐私保护的交易。简洁非交互式零知识证明的提出极大推动了这项技术在数字签名等方面的应用[2] 。zk-SNARK（Zero-Knowledge Succinct Non-interactive Arguments of Knowledge，零知识简明非交互式知识论证）指的是一种特定类型的零知识证明。它区别于一般的零知识证明在于其简洁性和非交互性。简洁性意味着证明的长度非常短，验证速度快；非交互性意味着证明者和验证者之间不需要来回交流。这使得证明者能够在没有事先交互的情况下说服验证者某个陈述是正确的，从而为区块链和隐私保护计算等场景提供了一种高效且可扩展的解决方案[3]。是一类应用广泛的通用零知识证明方案，通过将任意的计算过程转化为若干门电路的形式，并利用多项式的一系列数学性质将门电路转化为多项式，进而生成非交互式的证明，可实现各类复杂的业务场景的应用。目前，zk-SNARK已在数字货币、区块链金融等区块链领域实际落地，是目前最为成熟的通用零知识证明方案之一。

### 1.2.2 circom

circom是一种专门用于编写和优化零知识证明电路的框架，是学习zk-SNARKs的绝佳工具，其能够轻松地创建各种复杂的证明系统，使用声明式语言来描述电路，支持抽象化：并与SNARKs集成使开发人员能够轻松地将零知识证明应用到电路中。circom通过简化开发过程、提高效率和加速部署，为零知识证明电路的设计和应用提供了许多优势，有力地推动了这一领域的发展和应用[4]。

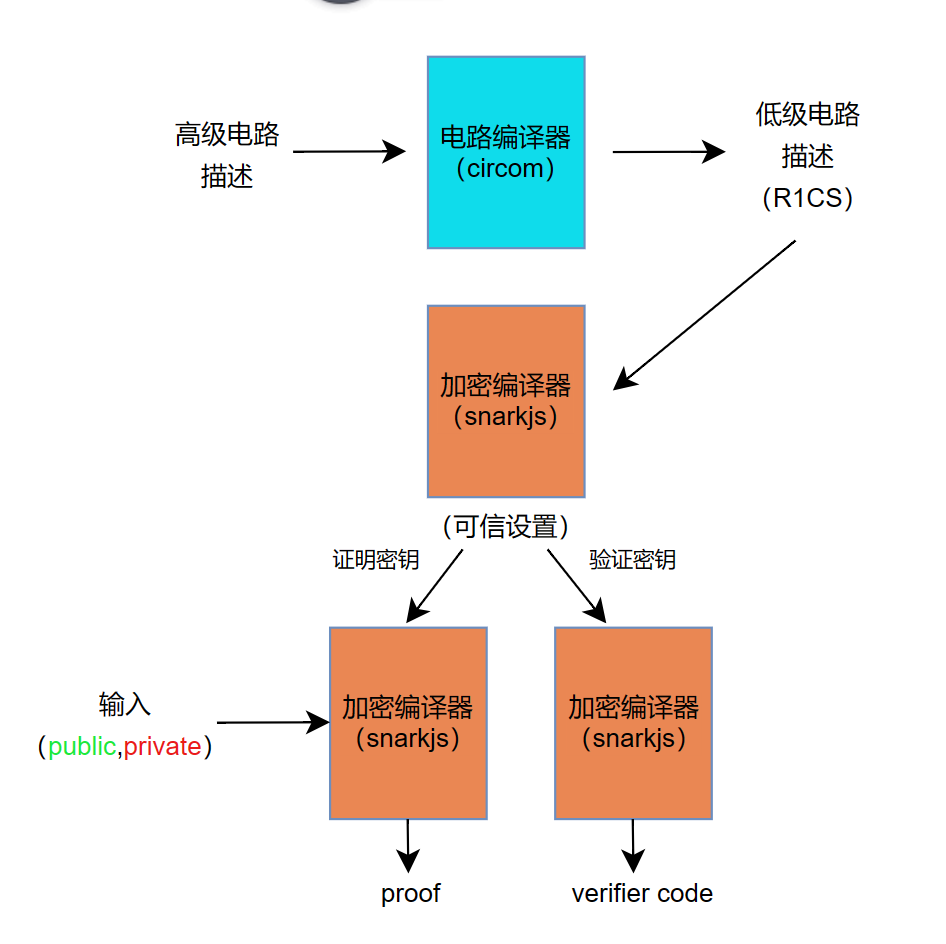


图4 circom结构[5]

### 1.2.3 Rescue-Prime算法

Rescue是一种哈希算法，旨在提供一种高度安全的哈希函数，尤其适用于零知识证明系统。Rescue算法采用了非线性的迭代结构，且算法设计中包含了大量的混淆和扰动操作，通过轮函数重复迭代来增加哈希函数的复杂度和安全性适用于零知识证明。Rescue算法的设计考虑了在零知识证明系统中的应用需求，具有适当的性能和安全性，使其成为零知识证明系统中的理想选择。更重要的是，Rescue算法在设计和规范方面采用了标准化的方法，以确保其安全性和可靠性，使其成为密码学领域的可信标准[8]。

而Rescue-Prime算法是Rescue算法的改进版本，专为零知识证明系统中的高效性而优化。它在几个关键方面进行了优化，包括算术表示的优化、专注于素数域而非二进制扩展域、简化的标准规范，以及针对算术复杂度的优化。Rescue-Prime哈希算法提供了高效率、强大的安全性保障，并且特别适合于零知识证明环境。Rescue-Prime通过以下几方面确保其安全性：

1. 抗碰撞性：Rescue-Prime设计中包含的复杂非线性迭代结构和混淆操作，提高了哈希函数的抗碰撞性，确保在合理的计算时间内难以找到两个不同的输入具有相同的哈希值。
2. 抗攻击性：Rescue-Prime在设计中充分考虑了抵抗已知攻击方法（如差分攻击和线性攻击）的能力，通过多轮函数迭代增加攻击难度，使其在理论上和实际应用中均表现出色。
3. 随机性：Rescue-Prime通过混合操作和置换增加输出的随机性，保证输出分布的均匀性，使其难以预测输出。

这些特性使得Rescue-Prime不仅适用于零知识证明系统，还在需要高安全性的应用场景（如区块链、智能合约和去中心化身份认证）中表现优异。总结来说，Rescue-Prime算法可以视为Rescue算法的一个特定应用版本，它继承了Rescue的一些特性，并针对零知识证明系统中的特殊需求进行了优化和调整。两者虽然有联系，但Rescue-Prime通过其特定的优化，提供了更适合于某些应用场景的特性。

### 1.2.4 circom实现的rescue哈希算法优势

circom是一种专门用于编写和优化零知识证明电路的框架，是学习zk-snarks的绝佳工具，其能够轻松地创建各种复杂的证明系统，使用声明式语言来描述电路，支持抽象化：并与SNARKs集成使开发人员能够轻松地将零知识证明应用到电路中。circom通过简化开发过程、提高效率和加速部署，为零知识证明电路的设计和应用提供了许多优势，有力地推动了这一领域的发展和应用[7]。

### 1.2.5 与其他哈希算法对比

①对比指标

1. Rank-1 Constraint System[6]

R1CS（Rank-1 Constraint System，一级约束系统）是一种用于表示计算的代数形式，适合用于zk-SNARKs（简洁非交互式零知识证明）。它将计算转换为一组线性约束，描述了输入变量和输出变量之间的关系。具体来说，R1CS通过以下形式来表达：

其中，、和是系数向量，是变量向量。每个约束表示为三个向量的内积，其结果必须满足上述乘积关系。

1. Algebraic Execution Time

AET (Algebraic Execution Time) 是一种用于评估STARK系统性能的指标。STARK是一种零知识证明协议，旨在提供高效且可扩展的证明系统。AET是在评估 STARK 性能时的重要指标之一，它衡量了在 STARK 系统中执行算术操作所需的时间。AET 包括以下几个方面的内容：算术运算的执行时间、电路复杂度、硬件性能的依赖性。较低的 AET 意味着系统能够更快地生成零知识证明，从而提高了 STARK 协议在实际应用中的可行性和效率。

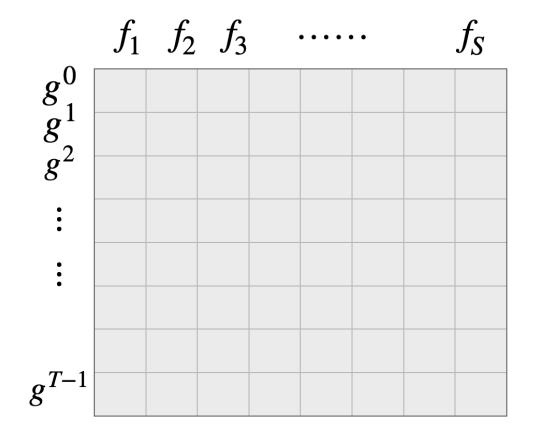


图5 Algebraic Execution Time:AET

②对比效果

**表1****各zk-friendly hash函数的效率对比**

|  |  |  |
| --- | --- | --- |
| 哈希算法 | R1CS(SNARK) | AET(STARK) |
| MiMC | 628 | 1944 |
| Poseidon | 276 | 425 |
| Vision | 360 | 576 |
| Rescue | **264** | **207** |

由表1可以得到：MiMC在R1CS中表现尚可，但在AET中效率较低，适合需要处理复杂多项式运算的场景。Poseidon在两种表示方法中都表现良好，适合大多数ZKP应用，特别是在需要处理大规模数据时。Vision适合硬件加速环境（如FPGA），在软件实现中效率不如Poseidon和Rescue。Rescue在两种表示方法中效率最高。

Rescue哈希函数是当代密码学领域的一项重要成就，以其高效的计算性能、强大的安全性和灵活的应用性质而备受瞩目[31]。特别是在互联网和数字技术飞速发展的时代背景下，Rescue哈希函数为保护用户隐私和数据安全提供了有力支持。

首先，Rescue哈希函数的设计优化了代数旋转[21]，以降低在生成哈希值时的计算量，从而在资源受限的环境中也能高效运行。circom是一种用于实现零知识证明系统的领先技术，它提供了一种简洁而灵活的方式来实现Rescue哈希函数。通过circom，使得可以轻松地构建出符合Rescue哈希函数规范的哈希算法，从而为各种应用场景提供了可靠的数据保护手段。

此外，Rescue是一种zk-friendly哈希函数，因其在零知识证明系统中的高效性而在区块链领域中获得广泛应用[7]。并且在智能合约中，尤其是在涉及加密和验证的合约中，Rescue可以用于哈希计算，它的高效性有助于降低计算成本，并提高合约执行的速度和效率；在去中心化身份认证系统（DID systems）中，Rescue可以用作哈希函数来生成和验证身份凭证，有助于提高系统的安全性和隐私性，同时确保身份验证过程的高效性；Rescue还可以用于构建Merkle树，使用Rescue作为哈希函数可以提高Merkle树的构建和验证速度，适用于各种区块链应用，如数据存储和验证；在去中心化存储系统中，如IPFS或Filecoin，Rescue可以用于哈希文件和数据块。这有助于确保数据的完整性和不可篡改性，同时提高存储和检索的效率。

Rescue-Prime是一族面向算术化的哈希函数，相对于原始的Rescue算法，它在特定上下文中更具效率。Rescue-Prime相对于Rescue算法的优势如下：

它是一种面向算术化的哈希函数，用于在密码协议中进行哈希计算，例如多方计算或零知识证明系统。

专注于算法复杂度：Rescue-Prime的设计目标是优化算术复杂度，而不是硬件或软件复杂度。

特定参数集：Rescue-Prime使用了特定的参数集，以适应STARK虚拟机（例如 Miden）的哈希需求。

素数域：Rescue-Prime使用了素数域，这在高级密码协议中通常优于二进制扩展域。

轮常数：Rescue-Prime使用基于SHAKE256的伪随机轮常数，增加了安全性。

优化：Rescue-Prime针对特定上下文进行了优化，适应了STARK虚拟机的需求。

总的来说，Rescue哈希函数以其高效的计算性能、强大的安全性和灵活的应用性质，在当代密码学领域具有重要意义。通过使用circom实现Rescue哈希函数，将其集成到各种应用场景中会变得轻松，为用户提供安全可靠的数据保护解决方案。随着互联网和数字技术的不断发展，相信Rescue哈希函数将会在更多领域展现出其巨大的潜力，为构建安全可靠的数字社会贡献力量。

在本作品中，采用circom实现零知识证明友好的Rescue-prime哈希函数，这不仅在理论研究方面具有重要价值，而且在实际应用中也展现了显著的潜力。从理论上讲，Rescue-prime哈希函数的设计符合零知识证明系统的需求，提供了较低的计算复杂度和高效的证明生成过程。其构造基于代数运算，有助于减少电路的门数，从而优化零知识证明的性能。此外，circom作为一种专门用于构建零知识电路的编程语言，能够有效地描述和实现复杂的密码学原语，使得Rescue-prime哈希函数的集成更加便捷和高效。

在实践层面上，使用circom实现Rescue-prime哈希函数能够显著提升零知识证明系统的整体效率。具体而言，circom编译器能够将高层次的电路描述转化为低层次的电路表示，优化后的电路在实际运行中表现出较高的性能。通过这种方式，Rescue-prime哈希函数在零知识证明中的应用变得更加实际可行，满足了对计算资源和证明时间的严格要求。此外，circom社区的活跃开发和持续改进，也为这一实现提供了坚实的支持和保障。

总的来说，本作品的工作在理论和实践两个方面都展示了通过circom实现零知识证明友好型Rescue-prime哈希函数的深远意义。这一实现不仅推动了密码学和零知识证明技术的发展，也为其在实际场景中的广泛应用奠定了基础。

## 1.3作品概述

本作品的核心任务是对各种零知识编程语言和哈希函数进行深入的调研和对比，最终选择使用circom+snarkjs工具对rescue-prime算法进行实现并进行全面的测试。在这个过程中，本作品首先对circom语言以及其他零知识编程语言进行了详细的探讨和对比，同时也深入研究了rescue-prime哈希函数在构造、效率、安全性等方面的优势。

为了使用circom语言实现rescue-prime哈希算法，本作品进行一系列的研究工作。首先，本作品深入研究circom语言的语法，理解了其独特的特性和使用方法。接着，对rescue-prime哈希算法的构造和运行流程详细分析，理解了其内部的工作机制。最后，本作品给出了rescue-prime哈希算法每个步骤和组件的R1CS算术电路形式表示，为最终的实现做好了铺垫。本作品的实现不仅展示了circom在构建零知识证明电路中的强大能力，也验证了Rescue-prime哈希算法在保护数据安全和隐私方面的巨大潜力。

为了测试rescue-prime哈希函数性能，本作品通过snarkjs运行其证明生成过程和验证过程。并同传统哈希算法如SHA-256和零知识友好型哈希函数rescue、poseidon进行对比。在测试过程中，本作品重点关注了算法的运行时间、circom文件编译约束数量、证明密钥大小等指标。这些指标反映了算法是否零知识友好、电路的复杂程度以及适用场景。通过精心设计的测试案例，本作品评估了算法在不同场景下的表现，确保它能够满足现实世界中对速度和安全性的严格要求。本作品的测试结果表明，Rescue-prime哈希算法在circom环境下运行稳定，且能够有效地集成到更广泛的加密系统中。

本作品的研究成果不仅为学术界提供了宝贵的参考，也为工业界实现更安全的加密解决方案提供了实用的指导。这一工作将为未来数字世界中的安全和隐私保护做出贡献。

## 1.4前景分析

随着信息技术的不断发展，数据安全和隐私保护变得越来越重要。Rescue 函数作为一种新兴的密码学算法，具有高度的安全性和数据完整性，因此在各种应用场景中的安全需求中将会越来越受欢迎。

在区块链中，rescue可以用于数据完整性验证[14]，哈希函数用于生成数据的唯一哈希值，确保数据在传输和存储过程中没有被篡改。每个区块的哈希值链表结构确保了整个区块链的完整性和安全。除此之外，其还可以用于交易验证，哈希函数可以快速验证交易的有效性。例如，比特币使用 SHA-256 哈希函数来生成区块头部的哈希值，以确保交易的不可篡改性[15]，而如果使用rescue prime可以更快更准确。另外，在工作量证明机制中，矿工需要找到一个满足特定条件的哈希值，这通过计算大量哈希值来实现，确保了区块链网络的安全性和去中心化，本作品中实现的函数可以达到更好的效果。

而在隐私保护中，rescue也可以起到重要作用[26]。首先是数据匿名化，哈希函数可以将敏感信息（如用户身份、交易记录等）转换成固定长度的哈希值，保护用户隐私。同时，哈希函数的单向性确保了原始数据无法通过哈希值反推出。在认证和签名中，哈希函数用于生成数据签名，确保数据在传输过程中的完整性和不可否认性。用户可以用哈希值验证签名，确保信息来源的真实性。更重要的是，在存储用户密码等敏感信息时，哈希函数可以将原始密码转换成哈希值存储，提高数据的安全性。即使存储介质被泄露，也无法直接获取原始密码。

此外，circom 作为一种专门用于零知识证明系统的领先技术，已经在区块链、隐私保护等领域取得了广泛的应用。利用 circom 实现 Rescue 函数将会使得该算法更加容易在各种应用中部署和使用，进一步推动其在实际应用中的普及和发展。

更重要的是，随着互联网和物联网的快速发展，大量的数据被生成和传输，对数据安全的需求也在不断增长[8]。Rescue 函数作为一种高效的哈希函数，能够有效地保护数据的完整性和隐私性，因此将会成为未来数据安全技术的重要组成部分。

# 第二章 作品设计与实现

## 2.1系统概述

从横向上看，本系统的体系结构可分为以下功能模块：

1. 算术电路模块：该模块使用circom语言实现Rescue-prime哈希函数电路。其中又可分为变量赋值模块和变量约束模块。它进一步分为变量赋值模块和变量约束模块。变量赋值模块负责将Rescue-prime算法的各个变量赋值给circom的零知识证明电路载体对象，而变量约束模块则根据赋值后的电路变量生成R1CS约束。
2. 变量赋值模块：将Rescue-prime算法的各个变量赋值给circom的零知识证明电路载体对象，其中变量可分为公共变量和私有变量。公开变量是指证明中的公开输入。另外，其运算过程分为模运算和位运算。位操作需要将变量拆分为多个位进行分配。当一个变量在计算中被多次使用时，不宜对其进行多次赋值，这样会造成变量的冗余和相同变量之间的重复约束。
3. 变量约束模块：该模块根据上一个模块分配的电路变量，即每个电路门的输入输出引脚所表示的值，生成R1CS约束。变量生成形式为(A, X)∗(B, X) = (C, X)的约束[9]。这也是最关键的一步；即生成了完整的算术电路结构。A、B、C为充分表示电路约束系数的矩阵，X为电路中变量构成的向量。只有正确的X才能使这个方程成立。最后生成一个电路文件。
4. 证据生成模块：在算术电路模块生成电路文件的基础上，证据生成模块负责创建可信设置，并生成证明密钥和验证密钥。算术电路模块实质上是要证明的问题的形式表示。这个模块还根据输入生成见证，然后证明人可以利用证明密钥、公开输入和见证来生成证据，证据是一个简短字符串，保证被验证者快速验证。这个过程涉及到复杂的数学运算和密码学原理，但通过snarkjs工具的帮助，这一切都被抽象化，使得开发者可以更容易地集成和使用零知识证明。本作品在此模块具体采用了Groth16协议，此协议在给定witness的情况下计算多项式系数以对应电路变量。随后选取两个随机数r,s，借助一系列运算生成证明。
5. 验证模块：验证模块中的验证方使用公开输入、证据和验证密钥来验证证据。这个过程确保了证明的有效性和证明过程的完整性。验证模块是整个系统的最终检验点，它确保了零知识证明的正确性和可靠性。本作品在该模块仍采用groth16，该协议在已知证明以及验证密钥的情况下，通过验证双线性配对等式是否成立，最终判断是否验证成功。

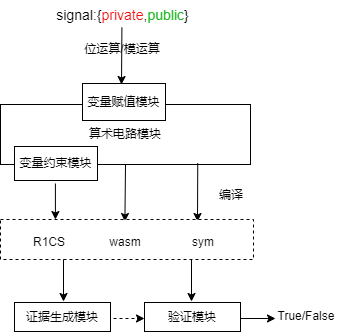


图6 系统模块

## 2.2系统需求分析

本系统旨在实现一个零知识友好型哈希函数——Rescue Prime，通过circom语言构建算术电路，以支持零知识证明的有效生成和验证。系统分为三个主要模块：算术电路模块、证据生成模块和验证模块。算术电路模块负责实现Rescue哈希函数的电路；证据生成模块负责创建可信设置，生成证明和验证密钥，以及根据输入生成见证；验证模块则用于验证证据的有效性。

### 2.2.1 功能需求分析

零知识证明的生成过程类似于编译器的原理。它可以分为两个部分：前端和后端。

(1)前端负责将需要证明的问题转换成可以被零知识证明协议接受的形式，如算术电路等。零知识证明电路生成是前端生成零知识证明中间形式的过程。在本作品中，使用circom高级编程语言实现Rescue哈希函数电路。具体步骤如下：

语法分析：解析 circom 高级语言语句，将其转换为中间表示。分析语句执行过程中的各个中间变量。

电路生成：将中间变量映射到算术电路的引脚变量。生成一个完整的算术电路，使其可以被零知识证明协议(如zk-SNARKs)接受。

(2)后端负责为特定的中间形式生成零知识证明。利用zk-SNARKs协议，后端的具体步骤如下：

生成协议参数：创建可信设置，生成公共参数和密钥对，包括证明密钥和验证密钥。

生成证明：根据前端生成的电路文件和输入数据，计算见证(witness)。使用证明密钥、公开输入和见证生成零知识证明。

验证证明：验证方使用验证密钥和公开输入验证生成的证明，确保其正确性。

综上所述，整体流程如下：

电路定义与编译：使用circom编写 Rescue 哈希函数电路。编译circom文件生成R1CS格式的电路描述。

创建可信设置：进行trusted setup，生成公共参数和密钥对。

计算见证：使用输入数据计算电路的见证。

生成零知识证明：使用证明密钥和见证生成证明。

验证零知识证明：验证方使用验证密钥和公开输入验证证明。

通过这种前后端分离的体系结构，系统能够高效地将复杂问题形式化，并生成和验证零知识证明。前端专注于问题的形式化表示和电路生成，后端则负责实际的证明生成和验证操作。这样不仅提高了系统的模块化和灵活性，还保证了高效的证明生成和验证过程。

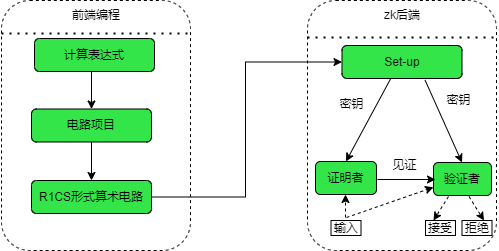


图7 功能需求

### 2.2.2 安全需求分析

零知识证明协议的安全性是系统设计的重要考量，它包括完整性、可靠性和零知识性三个方面。

完整性：系统必须确保所有的数据在传输和存储过程中保持完整，未被篡改或损坏。这意味着系统需要具备强大的数据校验机制，以验证数据的一致性和准确性。

可靠性：系统应保证在各种操作环境下都能稳定运行，即使在面临高负载或部分系统故障的情况下，也能保证服务的连续性和可用性。

零知识性：零知识证明的核心特性之一是证明者能够在不泄露任何除证明本身之外的信息的情况下，证明某个陈述的真实性。系统设计必须确保实现这一特性，使得验证者除了知道证明是正确的之外，无法获得任何额外信息。

验证零知识证明的有效性需要一种概率算法，该算法本质上是对零知识证明电路的所有电路门约束的挑战。只有当所有计算变换的R1CS约束都正确无误时，才能保证在输入证据有效的情况下，所有中间变量都满足约束。因此，设计需要保证约束生成的正确性。

对于零知识性，系统生成的证明必须确保包含的信息独立于电路的秘密输入信息，以保证证明过程中不会泄露任何不应公开的信息。

### 2.2.3 非功能需求分析

零知识证明协议的设计必须确保以下非功能需求：

1.正确性保障：协议设计需要保证R1CS约束的正确生成和验证，确保在输入证据有效的情况下，所有中间变量都满足约束。这包括严谨的约束设计、全面的测试和验证。

2.信息独立性：电路生成的证明必须独立于电路的秘密输入信息。通过引入随机性和信息隐藏技术，确保每个证明过程独立且不泄露任何敏感信息。

3.安全性验证：对协议的健全性和零知识性进行严格的安全性验证，采用形式化验证方法和概率算法，确保协议在各种可能的攻击场景下依然能提供可靠的安全保障。

4.效率优化：尽可能优化证明和验证的计算效率，以适应实际应用中的性能需求，同时确保安全性不被削弱。

5.可扩展性：设计具有良好扩展性的协议架构，以便在未来应对更复杂的计算和更多样化的应用场景。

通过满足上述非功能需求，零知识证明协议不仅能够确保其理论上的安全性和正确性，还能够在实际应用中提供可靠的隐私保护和高效的计算支持。

## 2.3技术背景

### 2.3.1 零知识证明发展与应用

零知识证明（Zero-Knowledge Proof, ZKP）的概念最早由Goldwasser、Micali和Rackoff在1985年的论文《The Knowledge Complexity of Interactive Proof Systems》中提出，旨在增强各种应用中的隐私和安全性，包括区块链技术。此后，ZKP的发展历史中出现了重要进展，推动了这一领域的进步。以下是一些具体的重要进展：

Fiat-Shamir Heuristic：这是一种将交互式证明系统转换为非交互式证明系统的方法，对ZKPs的实用化有着重大影响。

zk-SNARKs：由Alessandro Chiesa等人提出，这种非交互式、简洁的零知识证明协议在区块链技术中尤其受到关注。

Bulletproofs：由Benedikt Bünz等人提出，这是一种新型的非交互式零知识证明协议，它不需要可信设置，并且证明大小相对较小。

STARKs：由Eli Ben-Sasson等人提出，这种可扩展的透明的知识参数证明是zk-SNARKs的替代方案，不依赖于密码学的硬假设[12]。

Sonic：由Mary Maller等人提出，这是一种新型的零知识证明系统，可以实现任意大小的电路的递归证明。

这些协议和论文在提高ZKP的效率、降低计算成本以及扩展应用场景方面起到了关键作用。它们不仅加深了对零知识证明理论的理解，也为实际应用提供了强有力的工具。这些成就标志着ZKP从理论到实践的重要转变，为未来的技术革新奠定了基础。

ZKP的隐私保护属性正在改变Web工作的基础，其应用范围已经扩展到多个关键领域。在区块链技术中，ZKP提供了一种机制，使得交易双方能够在不透露交易内容的前提下验证交易的有效性，从而增强了交易的隐私性和安全性。在供应链管理中，ZKP允许审计员验证货物的运输和储存是否符合标准，而无需透露货物的具体信息，这对于保护商业机密和遵守监管要求至关重要。此外，ZKP在创建安全可验证的投票机制方面也显示出巨大潜力，它可以确保选民的隐私得到保护，同时保证投票结果的准确性和不可篡改性。

### 2.3.2 circom语言基础

circom是一种专门用于设计和实现零知识电路的编程语言。它简化了零知识证明电路的设计和实现过程。circom语言的主要特点包括：

1.模块化设计：允许用户定义可重用的子电路。

2.高效性：生成的电路可以高效地转换为具体的零知识证明系统。

3.灵活性：支持多种复杂的密码学操作和逻辑结构。

circom通常与snarkJS工具一起使用。snarkJS负责将circom编写的电路编译成中间表示（R1CS格式），并生成证明和验证所需的关键文件。开发者可以使用snarkJS在浏览器或其他JavaScript环境中加载这些文件，从而在这些环境中生成和验证零知识证明。

与Rust、Python等通用编程语言相比，circom在表示电路约束和生成ZK证明方面具有明显优势。circom的设计允许在多个处理器核心上并行计算，这使得算法能够充分利用现代计算机的多核性能，进一步提高电路的计算速度。与Halo2、Sage等其他专用于ZKP的工具相比，circom提供了更高层次的抽象和更易于使用的接口，使得开发者能够更快速地实现复杂的ZKP电路。circom 一直是 Dark Forest 和 Tornado Cash 等实际用例中突出的 ZK 应用程序的首选。它的受欢迎程度归因于其令人印象深刻的性能指标，具有优化的 WASM 证明带来的快速浏览器证明时间、通过 rapidsnark 的快速服务器端证明以及高效的链上验证。同时，circom 的兼容性集中在广泛使用的 ZKP 系统上，如 snarkjs 和 libsnark。

### 2.3.3 零知识友好哈希函数**Rescue-prime**

Rescue-Prime是一种专为零知识证明优化的哈希函数。

零知识友好哈希函数的概念最早可以追溯到1990年代，Nyberg和Knudsen提出了KN-Cipher，这是一种早期的尝试，用于在密码学协议中高效地评估哈希函数。随着零知识证明技术的发展，对于在有限域上运算高效的哈希函数需求日益增长。这些哈希函数需要在特定的数学领域内操作，如大素数或大二进制域，以适应零知识证明系统的要求。传统的哈希函数，如SHA-256和Blake2b，虽然在标准计算机硬件上表现出色，但在零知识证明系统中则显得过于复杂和缓慢。其中，基于HADES策略的poseidon和基于marvellous设计策略的rescue是其中的典型代表。

Rescue-Prime哈希算法是基于原始Rescue哈希算法的改进版本，它在几个关键方面进行了优化以适应特定的密码学应用。以下是Rescue-Prime相对于原始Rescue算法的主要改变：

优化的算术表示：Rescue-Prime专为在有限域上的算术运算进行了优化，这使得它在零知识证明系统中的评估更为高效。

选择素数域而非二进制扩展域：原始Rescue算法设计时考虑了在二进制扩展域上的应用，但Rescue-Prime专注于素数域，因为高级密码协议几乎普遍偏好素数域。

简化的标准规范：Rescue-Prime提供了一个简化的标准规范，去除了原始论文中不必要的复杂性和灵活性，使得算法更加直接和易于实现。

针对算术复杂度的优化：与原始Rescue算法相比，Rescue-Prime在设计时更加注重算术复杂度的优化，而不是硬件或软件复杂度。

与传统哈希函数相比，Rescue-Prime具有以下优势：

高效率：Rescue-Prime在零知识证明环境中的计算效率较高，适用于资源受限的环境。

安全性：提供了强大的安全性保障，能够抵抗多种已知的密码学攻击。

零知识友好性：设计时充分考虑了零知识证明系统的需求，能够在生成证明和验证过程中发挥最佳性能。

Rescue-Prime采用了置换网络SPN和代数结构，使其在保证安全性的同时，能够高效地进行哈希计算。其独特的设计使其在需要高效证明和验证的应用场景中表现出色，如区块链中的隐私保护和智能合约验证等。

## 2.4预备知识

### 2.4.1 符号定义

本章中所使用的所有符号定义如表2所示。

在零知识证明中，令来表示某一事件发生的概率。令表示一个断言，即证明者试图向验证者证明的命题。令代表一个任意小的正数，用以描述错误验证的概率无限小。令表示零知识证明中的见证（秘密信息）。 表示行列的矩阵。 表示算法以为输入、为随机输入生成的过程，表示函数映射关系。

本章中的算法输入均包含安全参数。 如果对于任意的多项式都存在常数 使得当时有，则称为对于的可忽略函数 。 记当。用PPT表示概率多项式时间 (probabilistic polynomial time)。记为省略安全参数多项式因子的大记法, 本文常省略。

本作品在R1CS以及零知识哈希函数介绍部分，令表示所有可能的二进制比特流。令表示所有可能的长度为的二进制比特流。代表一个有限域，这是一种数学结构，在其中可以进行加法、减法、乘法和除法运算而结果仍然在该集合内[15]。特别地，表示素数域，即所有模的整数构成的集合，其中是一个素数。采用小写粗体字母表示向量，例如，均是R1CS表示中所采用的系数向量。

表2符号定义

|  |  |  |  |
| --- | --- | --- | --- |
| **符号** | **含义** | **符号** | **含义** |
|  | 某一事件的概率 |  | R1CS表示形式中的系数向量 |
| *x* | 零知识证明中的断言 |  | 零知识证明中的见证（秘密） |
|  | 任意小的正数 |  | 有限域 |
|  | 所有可能的二进制比特流 |  | 某个素数 |
|  | 所有可能的长度为（固定常数）的二进制比特流 |  | 素数域 |
|  | 标量乘法 |  | R1CS中的向量点积/有限域乘法 |
| *λ* | 安全参数 |  | 行列的矩阵 |

### 2.4.2 零知识证明

本小节介绍零知识证明, 并给出若干相关概念的简要定义.

给定二元关系，记语言为集合。称一个语言是NP语言当如下两个条件成立：

(1) |*w*| = poly(|*x*|) 。

(2) 给定任意的 *x*、*w*, 存在多项式时间算法能够高效判定 。

记〈A, B〉为一对交互式图灵机 。 记〈A(y), B(z)〉(*x*) 为在 A、B 的随机输入带均匀独立选取, 公共输入为 *x*, A 的辅助输入为 *y*, B 的辅助输入为 *z* 时, 图灵机 B 与图灵机 A 交互后输出的随机变量。

零知识证明（英语：zero-knowledge proof）或零知识协议（zero-knowledge protocol）是证明者（P）向检验者（V）证明某命题的方法，特点是过程中除“该命题为真”之事外，而不会泄露任何有关witness（）的信息。因此，可理解成“零泄密证明”。 零知识证明提供了一种解决方案，它基于复杂的数学构造，如同态隐藏和交互式证明系统。

零知识证明四种性质及其形式化表述如下：

•完备性（complete）

若所要证之事为真，则诚实（意即依协议行事）的证明者能说服诚实验证者。

•正确性（sound）

若命题为假，则作弊证明者仅得极小机会能说服诚实验证者该事为真。

•零知识（zero-knowledge）

若命题为真，则验证者除此之外，过程中没有得悉任何其他资讯。换言之，仅知命题为真（而不知秘密本身）已足以“想像”出一个交互的情境，其中证明者的确知道该秘密。

其中， L 表示语言集合，表示对于*x*的所有可能的见证集合， Pr 表示概率，表示一个非常小的正数，View 表示验证者的视图，S表示模拟器， 表示计算上不可区分，右上角带有C表示集合的补集。

零知识证明可以是交互式的，也可以是非交互式的。在交互式零知识证明中，证明者和验证者之间会有多轮的交流。而非交互式零知识证明（如zk-SNARKs）则允许证明者生成一个证明，验证者可以独立地验证，无需与证明者交互。零知识证明广泛应用，包括对于私人数据的证明说明、匿名授权、匿名支付、外包计算等等。

最流行、最高效、通用的零知识协议是零知识简洁非交互式知识论证（ZK-SNARK）。与传统的零知识证明相比，zk-SNARK有以下特点：

•简洁(succint)——独立于计算量的证明是恒定的，小尺寸的。

对于一个交互式论证, 如果 P 和 V 之间的通信复杂度不超过 , 则称该论证 〈P, V〉 为简洁 (succinct) 的; 如果通信复杂度不超过 ,则称论证为略显简洁的 (slightly succinct), 其中 c 为某个小于 1 的常数 。 本文将简洁和略显简洁的零知识证明统称为简洁零知识证明。

•非交互式(non-interactive)——一旦证明被计算出来，它就可以用来说服任何数量的验证者，而无需与证明者直接交互。

•有论证的知识(with argumented knowledge)——陈述是正确的，具有不可忽略的概率，即虚假的证明是不可构建的;此外，证明者知道真实陈述的对应值，例如，如果陈述是“是的结果”，那么证明者知道一些a，使得，这是有用的，因为B只能用a的知识来计算，并且仅从B计算a是不可行的。

以下为该性质的形式化表述：

知识论证 (Argument of Knowledge) 指具有知识可靠性的论证 。 给定一个多项式时间内可判定的二元关系 R 及其对应的 NP 语言 L(R), 知识可靠性是指对于任意的 PPT 敌手, 都存在期望 PPT 的提取器 E, 使得对于 L(R) 及任意的陈述 *x*、，如果有, 其中 p(·) 为某个多项式, 则有 Pr[← (*x*) : R(*x*, ) = 1] ≥1/p(|*x*|) − negl(|*x*|) 。

以上特点使得它在区块链和其他需要高效隐私保护的应用中变得非常有价值。

zk-SNARK的大致流程如图8：

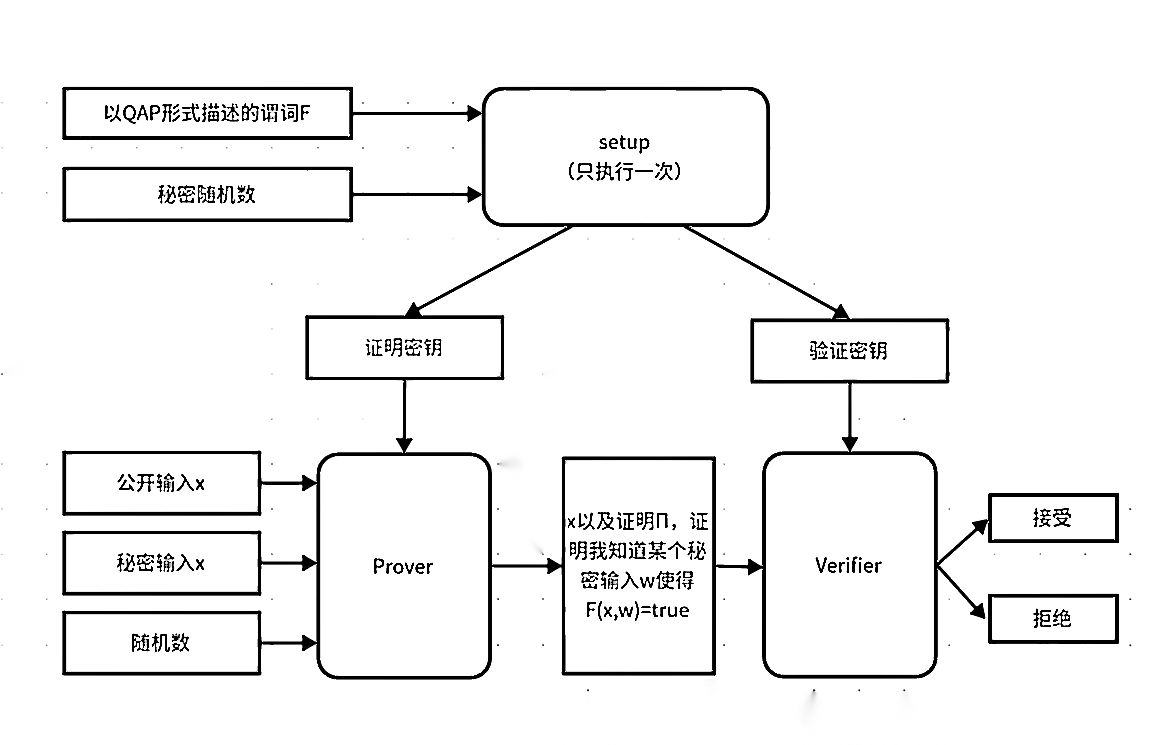


图8 zk-snark的大致流程

zk-SNARK的核心是一种特殊的证明系统，它可以生成非常小的证明大小，并且验证过程非常快速。这是通过一系列复杂的数学构造实现的，包括同态加密、多项式插值和椭圆曲线密码学[43]。目前常见的zk-snark协议，包括Pinocchio协议、Groth协议等等。

与大多数 ZK 系统一样，ZK-SNARK 在算术电路模型中运行。当用户需要验证他们的信息时，他们会将其转换为计算问题。为了在计算机系统中结构化地表达计算，zk-SNARK 选择了一种有效的中间体结构，有限域上的算术电路。 具体来说，算术电路由一系列逻辑门组成，组合基本运算的构建用于各种计算任务的复杂算术电路。

零知识证明电路是一个重要的概念，是通用零知识证明协议的关键步骤。一般的零知识证明协议是基于电路可满足性的，它可以将任何NP问题转化为零知识证明。

**定义1**（算术电路）算术电路是一个的映射C，其中 F 是有限域。

(1)它是一个有向无环图 (DAG)，其中内部节点标记为 + 或 ×，输入标记为，边是电线或连接。

(2)它定义了一个带有评估公式的 n 变量多项式。

(3)其中 |C| 表示映射 C 中的门数量。

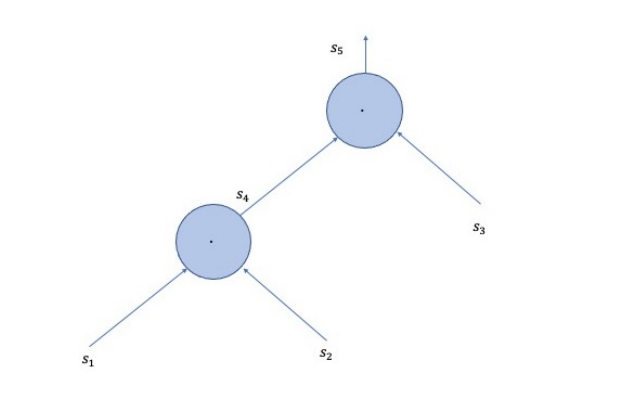


图9 算术电路示例图

**定义2**（电路可满足问题）电路可满足问题 (Circuit Satisfiability Problem, C-SAT) 是指给定电路C、电路的部分输入(可为空) 和电路输出y，判断是否存在证据***w*** (电路的另一部分输入, 视为秘密输入) 使得C(x, ***w***) = y。如无特殊说明, 本作品中的零知识证明均是针对 C-SAT 问题的。针对布尔电路可满足问题的零知识证明可通过调用针对算术电路可满足问题的零知识证明高效构造(只需扩大域并增加对变量为 0 或 1 的约束即可), 反之, 尚不清楚是否有高效的转化方式。

### 2.4.3 R1CS（一阶约束系统）算术电路介绍

R1CS 全称 Rank-1 Constraint System，一阶约束系统，是一种常见的算术电路格式，其本质是一个方程组。它在零知识证明（Zero-Knowledge Proofs, ZKPs）和其他密码学应用中扮演着核心角色。R1CS 描述了一组基于约束的计算规则，可以使用一组公共参数和私有输入序列来验证这些规则。该模型将计算表示为一组约束条件，即线性方程和不等式。每个方程都有自己的一组系数，而每个变量代表一个输入或输出值。这些方程和不等式总体上描述了计算的限制条件，这意味着满足这些条件意味着可以针对给定的输入序列正确计算相应的输出结果。

**定义3** （一阶约束系统，rank-1 constraint system, R1CS）是七元组, 其中表示公共输入输出向量, ，, n 是所有矩阵中非零值的最大数目。称 R1CS 问题是可满足的当且仅当对于一个 R1CS 组, 存在证据使得, 其中。

记 n 为 R1CS 可满足问题的规模, 可以证明，任意 C-SAT 问题都可用 R1CS 可满足问题表示,且n = O(|C|)。R1CS 是高级语言编译器的常见目标程序且形式较简单, 同时任意 C-SAT 问题都可用 R1CS可满足问题表示, 故有一些零知识证明 在应用实现时是先将 C-SAT 问题转化为 R1CS 可满足问题, 再针对 R1CS 可满足问题构造的; 也有部分零知识证明直接针对 R1CS 可满足问题

其中 、 和 是系数向量 ( 由对某个素数取模的整数组成的素数域的元素），并且是不同“伪变量”的向量。 每个伪变量要么是一个变量，表示域的一个元素，要么是特殊符号 1，表示域元素 1。· 表示取两个向量的点积，但所有加法和乘法均模p， × 表示 2 个标量的乘积模p。 使用伪变量 1 允许用点积表示常数加数。

R1CS 中的每个向量三元组表示在算术电路中构造的数学约束。 这些向量包含解向量***s***中相应位置处的变量系数。 R1CS 的解向量 ***s*** 包括 R1CS 方程中出现的所有变量的赋值，它是的解。

R1CS是一个强大的计算模型，广泛应用于许多实际应用中。 作为可验证算法设计的重要组成部分，它极大地扩展了人们对计算机科学和密码学的理解，并为各种隐私保护措施提供了关键支持。 通过对变量约束的形式和排序施加约束，提出了约束组的 R1CS 范式。

**定义4**R1CS范式是满足以下要求的R1CS：

1. 如果R1CS范式中的约束包含变量之间的乘法，则它不能有任何其他运算符。
2. 如果R1CS范式中的约束不包含变量之间的乘法，则它不能包含其他线性约束生成的中间变量。
3. R1CS范式中约束和变量的排序（定义3中定义）必须与本作品的排序方法（×在+前面）一致。

对于任何计算，都有输入值和输出值。当证明者知道使计算产生特定输出值的输入值时，他必须知道计算过程的任何中间值。使用算术电路表示计算过程以便计算机处理尤为必要。每个电路门对应的两个输入值和一个输出值需要满足相应的约束。R1CS矩阵正适应描述转换电路的所有门约束的要求。通过该矩阵可以生成相应的证明多项式，并根据概率多项式验证的性质对zk- snark进行证明和验证。在零知识证明中，需要人工实现的部分是将计算转化为矩阵形式的电路约束。

### 2.4.4 零知识友好型哈希算法

哈希函数能将任意长度的输入转换成一个固定长度的输出，这个固定长度的输出被称为原消息的哈希值，理想的安全的哈希函数要求是满足抗碰撞性的。哈希函数可以表示为一个映射。

其中表示所有可能的输入数据集合，而表示固定长度为 n的输出数据集合。

理想的哈希函数要满足以下安全特性：

(1)弱抗碰撞性：对于给定的*x*,找到*y=x*使得H(*x*)=H(*y*)是困难的

(2)强抗碰撞性：找到任意的*x*=*y*使得H(*x*)=H(*y*)是困难的

(3)单向性：对于给定的哈希值h,找到*x*使得H(*x*)=h是困难的

哈希函数在零知识证明中也有广泛应用，比如完整性检查和签名等。传统标准化哈希函数比如SHA-2和SHA-3虽然安全性得到普遍信赖，但它们在零知识协议中并不一定有很高的效率。造成这种情况的主要原因是 ZK 协议的效率是 以与传统指标完全不同的方式确定，例如 运行时间、功耗和门数。效率零知识协议中的电路取决于它们的代数结构[11]。

通常，如果电路表示为大域中的简单表达式，它可以实现执行时间和证明规模上的高效证明。不幸的是，传统的哈希函数不适合这样做。

零知识友好型哈希算法（ZK-friendly hash functions）是哈希算法的一个特殊类别，它们在保持上述哈希函数特性的同时，还特别优化了在零知识证明系统中的应用效率。这类哈希算法在构造零知识证明时，能够有效减少所需的计算资源和时间，使得证明过程更加高效[14]。它们通常有以下步骤：

有限域运算：ZK-friendly哈希函数通常在大的有限域（例如GF(p)）中进行运算，这有助于简化证明过程中的代数结构。

简化的电路设计：这些哈希函数的设计减少了乘法门的数量，从而降低了证明生成的复杂性。同时放弃采用传统哈希函数使用的复杂位运算。最为杰出的设计是S盒，通常是低次幂运算，在安全参数与SHA256相同的情况下，它将电路大小减少了两个数量级。

海绵结构（Sponge construction）：一些ZK-friendly哈希函数采用海绵结构，这种结构没有密钥调度，分析起来更简单。海绵结构的基本构造如图10：

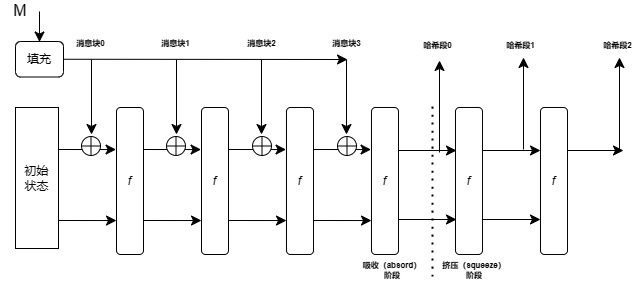


图10 海绵结构的基本构造

零知识友好型哈希算法在区块链项目中得到广泛应用，如Filecoin的Merkle树证明，Loopring在以太坊第二层的私有交易等[13]。

## 2.5关键设计

### 2.5.1 电路层次结构设计

根据电路规模和调用关系，本作品将Rescue-Prime零知识证明电路垂直层的体系结构设计分为若干层。底层电路模块封装后，底层模块可以被高层模块直接接触，下层对上层透明[16]。它的四层结构从下到上分为：辅助操作层、核心操作层、迭代压缩层、主函数层。

电路的垂直层设计如图11所示，每一层电路都由几个子电路模块组成。

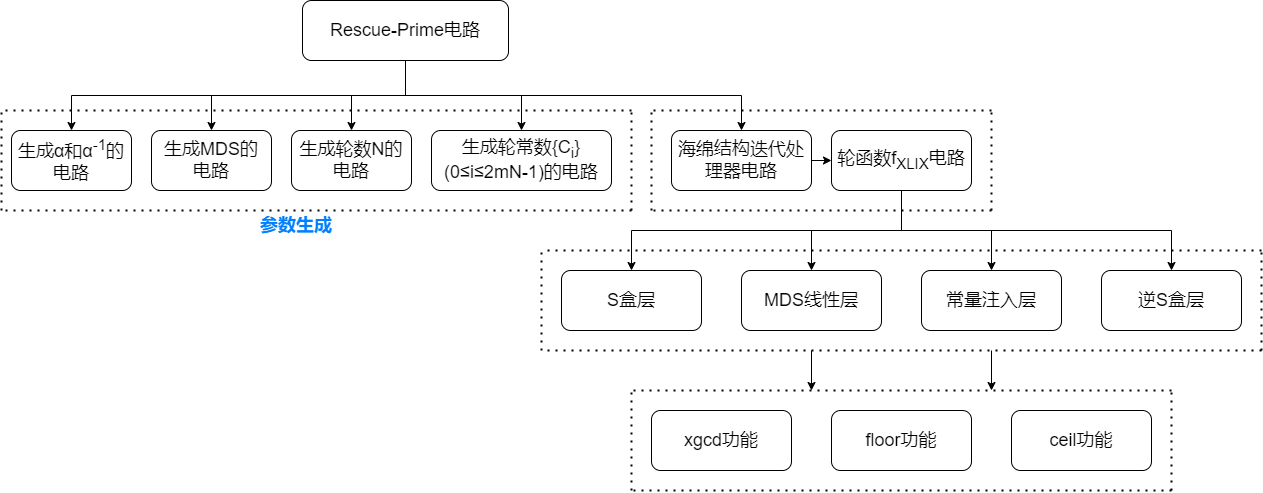


图11 电路的垂直层设计图

（1）辅助操作层

该层提供本设计所需的基本操作功能函数，可实现扩展欧几里得算法、取最小值、取最大值等基本操作，为上层电路实现提供必要支持。

①xgcd功能函数：扩展欧几里得算法用于计算两个整数和的最大公约数(gcd)，以及系数和使得*)*。

②floor功能函数：返回小于或等于给定实数的最小整数。

③ceil功能函数：返回大于或等于给定实数的最大整数。

④及其逆元生成函数，以质数p作为输入，通过循环遍历得到满足需求的指数，再利用扩展欧几里得算法计算得到其逆元。

⑤组合数以及阶乘函数bominal、factorial。

⑥轮数生成函数，用于产生控制哈希函数中的rescue-XLIX-permutation置换次数的参数。

（2）核心操作层

这一层提供了Rescue-Prime哈希函数中最核心的海绵结构压缩的算术电路。Rescue-Prime的压缩函数涉及到多轮迭代操作，每轮迭代操作都涉及到该层中的S盒变换、MDS矩阵乘法、常量注入和逆S盒变换。下述功能模块共同构成了Rescue-Prime轮函数的核心算法Rescue-XLIX置换[42]，如图12所示。

①S盒层：对状态的每个元素进行幂运算的映射；

②MDS线性层：使用MDS矩阵对状态进行矩阵向量乘法；

③常量注入层：将轮常数列表中的下一组个常数加入到状态中；

④逆S盒层：对状态的每个元素进行逆幂运算的映射。

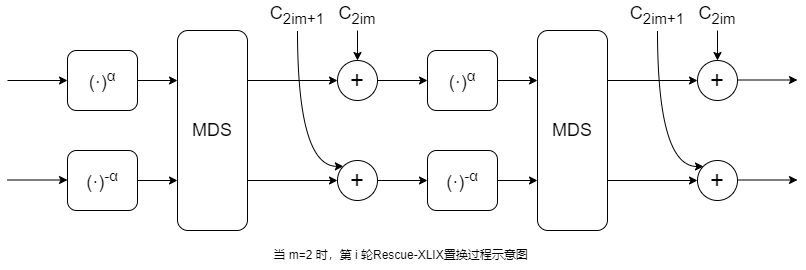


图12 第 i 轮Rescue-XLIX置换过程示意图

（3）迭代压缩层

该层实现Rescue-Prime的海绵结构迭代过程，包括吸收和挤压两个阶段，如图13所示。在吸收阶段，输入数据被分块并逐块处理；在挤压阶段，处理后的数据被逐块输出。这一层的设计使得Rescue-Prime哈希函数能够处理任意长度的输入数据 。

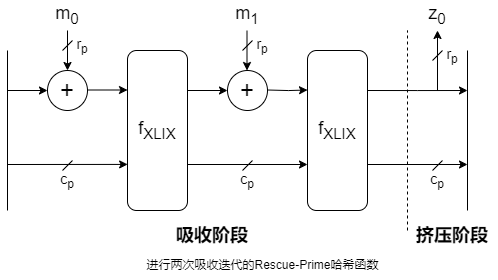


图13 Rescue-Prime的海绵结构迭代过程示意图

（4）主函数层

该层将参数生成功能模块和迭代压缩功能模块整合起来，从而实现对消息输入计算Rescue-Prime哈希值并生成电路约束。这一层是整个电路的顶层，它直接面向用户，提供了整个电路的接口

本作品的电路分层设计是根据Rescue-Prime哈希算法对所有计算步骤进行转换。因此，设计从计算过程和计算形式两个维度保证约束生成的正确性。

零知识应该保证电路生成的证明包含独立于电路秘密输入的信息。在circom语言中编写电路后，可以通过一系列步骤将其编译为WebAssembly (WASM)文件来生成证明，这个证明包含了独立于电路秘密输入的信息，可以在浏览器或其他支持WASM的环境中验证电路计算的正确性。

如前所述，该电路在理论设计上是安全的，在第三章的功能测试中可以进一步验证其正确性。

2.5.2 rescue-prime安全性分析

首先从整体上分析：

对于零知识证明协议来说，其安全性由三部分组成:完整性、稳健性和零知识。在这里，将其完备性和稳健性结合起来作为正确性，并将其与零知识分开分析。这两个属性确保了，只有当证据有效时，所有的中间变量都满足约束。

验证零知识证明需要一种概率算法，这在本质上挑战了零知识证明电路的所有电路门约束。只有计算变换的所有R1CS约束都是正确的，才能保证在输入证据有效时，所有中间变量都满足约束。

本作品的电路设计采用了分层的方式，这是根据Rescue-Prime哈希算法对所有计算步骤进行转换的结果。这种设计从计算过程和计算形式两个维度保证了约束生成的正确性。这意味着，不仅要关注计算的结果，还要关注计算的过程。

零知识应该保证电路生成的证明包含独立于电路秘密输入的信息。在circom语言中编写电路后，可以通过一系列步骤将其编译为WebAssembly (WASM)文件来生成证明，以在浏览器或其他支持WASM的环境中验证电路计算。

下面对算法本身进行安全性分析：

从rescue-prime算法来看，其安全性在以下集中攻击方式下是安全的，分析如下：

1. 线性密码分析：线性密码分析是一种常见的密码攻击方法，它试图找到密码算法的线性近似来预测输入输出位关系。因此，如果一个算法能够有效地打破明文、密文和密钥之间的线性关系，或者使得这种关系难以找到，那么它就可能能够抵抗线性密码分析。现有研究通过对Rescue进行了线性密码分析，通过使用Fp上的加性字符的概念，发现单一线性路径对Rescue并不构成威胁。对于线性路径聚类也未发现Rescue存在弱点。
2. 差分密码分析：差分密码分析是另一种常见的密码攻击方法，它试图通过比较输入和输出的差异来找到密码算法的弱点。因此，如果一个算法能够确保即使明文有微小的改变，密文也会有大的变化（这种属性被称为扩散），那么它就可能能够抵抗差分密码分析。现有研究对于Rescue进行差分密码分析，探索了差分路径如何在Rescue中传播，可以看到在这种情况下也没有发发现任何问题。
3. 乘法子群传播：乘法子群传播是一种新的攻击方法，它研究了循环函数如何传播乘法子群。它基于乘法子群的特性来进行攻击。在有限域上，乘法子群的阶必须是素数域的阶的因子。因此，如果一个算法能够确保其乘法子群的阶具有大的平滑性，那么它就可能能够抵抗乘法子群传播攻击。通过研究循环函数如何传播乘法子群发现了与密钥调度相关的性质，这也在一定程度上反映了Rescue的安全性。
4. 反弹攻击：反弹攻击是一种针对哈希函数的攻击方法，它在寻找碰撞方面的应用非常有效。这种技术可以有效地找到特定差分路径的解决方案，通常用于寻找碰撞或区分器。因此，如果一个算法能够确保即使在中间轮次注入差分，前后轮次的差分传播也会被有效地混淆和扩散，那么它就可能能够抵抗反弹攻击。对于Rescue-prime算法，反弹攻击也被认为是安全的。
5. 抗量子攻击：在量子计算的威胁下，密码算法的抗量子攻击能力变得越来越重要。Rescue-Prime 在使用任何 MDS（最大距离分隔）矩阵时都是安全的。这意味着，例如，Polygon Zero 提出的一种循环 MDS 矩阵可以被用于Rescue- Prime的实例化，从而提供一种安全的密码学解决方案。这种灵活性和安全性是 Rescue-Prime 在抗量子密码领域的优势之一。
6. 安全参数选项：Rescue-prime提供了安全参数选项，用以调控哈希算法轮数。这为用户提供了一种灵活的方式来平衡安全性和效率。
7. 多种自选参数：Rescue-prime提供了多种自选参数，保证了哈希算法的灵活性和可扩展性。用户可以通过灵活调整这些参数来保证算法的安全性。

如前所述，该电路在理论设计上是安全的，在第三章的功能测试中可以进一步验证其正确性。

2.5.3 零知识证明电路的实现

在密码学中，哈希算法的实现和优化是一项重要的工作。本作品的核心部分和创新点在于，针对rescue-prime的零知识证明（zk）电路实现的难度，提出了一种新的实现方法。当前circom的开源库circomlib中有已对多数其他零知识友好哈希函数如mimc、poseidon、pedersen进行了circom语言实现。circom作为目前最流行的zk证明系统，缺少这一部分的构建。

其实现难度有以下几点：

1.circom语言本身的难度

circom语言引用了template关键字，用于实例化电路。易与一般语言的函数造成混淆。circom语言存在信号signal和变量var的区别，signal不允许重复赋值。circom需要定义约束式，约束式需要满足R1CS，造成高次幂运算的困难。circom语言保留关键字相比常用编程语言极少，缺少终结循环的break。同时circom缺少数学运算库。开源库circomlib仅提供数字转比特串等简单运算。大量数学函数比如floor、ceil需要自主实现。另外由于circom语言较为小众，关于circom的报错信息难以查找，同时circom编译器的报错信息存在偏差，影响编程人员的判断。

2.Rescue-prime多元组件和流程

Rescue-prime定义了大量参数生成方法，包括轮数生成、S盒幂次选取、MDS矩阵生成、轮常量生成。需要大量基础数学运算比如阶梯化矩阵、扩展欧几里得算法等等。Rescue-prime逆S盒步骤巨大的也对其circom实现增加难度。

3.对R1CS须有了解

R1CS一定程度上是为了将各种问题转化为计算机理解的形式。与人类理解问题的方式存在一定冲突。

4.非二次约束处理

在Circom中，每个约束都必须是二次的。这意味着如果有一个包含多个乘法的约束，需要将其拆分为多个二次约束。这可能会增加电路的复杂性，并需要额外的编程技巧。

本作品的创新点在于一定程度上克服以上难题：

1.过于复杂的操作采用function而非template实现

使用template意味着需要定义大量signal，signal的赋值往往需要通过约束式，使用<==、==>运算符，令电路更加复杂，而采用function可以避免这种问题。

2.采用算法生成参数而非事先确定参数

在已有的哈希算法circom语言实现中，如poseidon.circom、mimc.circom中大量参数如S盒幂次、轮常量等已经预先确定。这无疑降低了算法的灵活性。而本作品选择对S盒幂次等参数生成进行保留，以保证算法的灵活性。

3.使用快速模幂思想提高逆S盒步骤效率同时解决欠约束问题

在已有的高次幂运算circom实现中，大多采用的形式。

其中是一个circom中的信号类型变量，*num*是底数，*pow*是指数，是circom语言对信号的赋值符号，\*\*代表circom语言中的幂运算符。

这种形式的问题在于使用 <-- 运算符赋值时，并不会自动生成约束。如果需要添加约束，应该使用 <== 运算符。这就会引发circom语言中的欠约束问题，导致电路的行为不完全确定，进而引发电路输出不唯一。这可能会影响电路的正确性、安全性和可靠性。而在零知识证明系统中，欠约束可能导致不正确的证明，从而影响安全性。

而circom的低次幂运算通常开一个信号数组存储每轮幂值，

其中*base*是底数。

这种方法虽然解决了欠约束问题，但会引起数组过大，效率低下等问题。Rescue-prime逆S盒步骤的指数通常会超出无符号正整数类型的值。

于是设法将快速模幂应用到circom语言中。具体见6.(4)逆S盒约束。

总的来说，本作品通过以上创新点，成功地克服了circom语言和rescue-prime算法实现的难度，为rescue-prime的zk电路实现提供了一种新的可能。这无疑对于推动rescue-prime在zk证明系统中的应用具有重要的意义。在未来，期待看到更多的研究和应用来进一步提升这个算法的性能和安全性。

本节将Rescue-Prime每一步的具体操作转换为加法门和乘法门，生成R1CS约束关系。为可以生成具体的R1CS约束条件，本节内容以固定参数的一轮运算为例。对于目前的SNARK证明，使用以下参数：

①生成素数域的大素数

②S盒步骤的幂：，

逆S盒步骤的幂：

③哈希函数的状态宽度：

④

在上述参数的作用下，生成的其他参数如下：

①该算术sponge的rate，决定了Rescue-XLIX permutation之间absorb的field elements数量：

②轮数

③最大分离矩阵（用以对信息作充分的扩散）：

④轮常数（前2个常数值示例）：

某一轮中的状态向量表示为。

1.运算功能以及参数生成函数

(1)xgcd函数：可实现通过扩展欧几里得算法计算元素的逆元；

(2)floor函数：可实现取输入元素中的最小值；

(3)ceil函数：可实现取输入元素中的最大值。

(4)及其逆元生成函数，以质数p作为输入，通过循环遍历得到满足需求的指数，再利用扩展欧几里得算法计算得到其逆元。

(5)组合数以及阶乘函数bominal、factorial。

(6)轮数生成函数，用于产生控制哈希函数中的rescue-XLIX-permutation置换次数的参数。

2.核心运算电路

(1)S盒层约束

在S盒变换的运算中，实际进行的是幂操作。在上述参数的实例中，已知一个状态中有两个域元素，对这两个域元素中的第一个元素取次幂的运算过程如下：

以状态中第一个元素为例，该过程包含两个R1CS约束。

第一个R1CS约束的三个变量项为：

第二个R1CS约束的三个变量项为：

(2)MDS线性层约束

该运算过程为使用MDS矩阵对状态进行矩阵向量乘法，即。具体计算如下：

即：

引入中间变量来表示每一步的中间结果：

然后，有

生成的R1CS约束如下：

*1.：*

2.：

3.：

4.：

5.：

6.：

(3)常量注入层约束

以首次循环中的第一次常量注入为例：

第一个R1CS约束的三个变量项为：

第二个R1CS约束的三个变量项为：

(4)逆S盒层约束

与S盒运算过程一致，均为幂运算。但本部分采用了快速模幂思想：

|  |
| --- |
| **Algorithm 1** 快速模幂算法 |
| **Data:** 底数, 指数, 模数  **Result:**  **if****then**  **return** 1  **end**    **while****do**  **if****then**  **end**  **end**  **return** |

首先需要定义一个num2bits函数以将指数*pow*转化为二进制比特串*bits*，便于后续操作。这个函数的工作原理是通过逐位操作，将指数从低位到高位的二进制表示提取出来。

然后定义一个*powers*信号数组，这个数组的作用是临时存储在计算过程中产生的中间结果。定义为数组是考虑到circom中信号类型不允许重复赋值。随后定义的二维数组下标中的*r*均代表当前轮数。

再定义一个*temp*二维数组，

通过定义数组和数字转比特流操作，成功克服了circom中信号不允许重复赋值的困难，在circom中实现快速模幂，同时提高效率并解决了欠约束问题，有效保证了算法的正确性和安全性。

3.迭代压缩电路

经过上述核心运算组成的轮函数之后，假设下一个状态，则对于当前状态经过S盒幂运算、MDS矩阵乘法运算、轮常数加法运算、逆S盒幂运算、MDS矩阵乘法运算、轮常数加法运算之后输出，赋值运算如下：

第一个R1CS约束的三个变量项为：

第二个R1CS约束的三个变量项为：

4.主函数电路

(1)填充预处理

首先接受输入序列并将其填充到合适的长度，以满足后续处理的需求。对输入序列添加分隔符(1)和填充位(0)：

其中代表填充的长度，代表输入序列长度,代表填充后长度，是填充后序列，是原始输入序列，代表原始输入序列的第*i*个元素，同理。

(2)吸收阶段

每一轮的状态更新可以用以下R1CS约束来表示：

其中*j*代表轮序号。

(3)挤压阶段

最后，挤压（squeezing）阶段的输出序列由最后一轮的状态决定：

=

最后本节尝试借鉴Rescue-prime Optimized。由于Rescue算法以安全作为其首要目标，其在CPU或者FPGA上运行时仍具备提升空间。Rescue-prime Optimized在STARK中具备与Rescue-prime相同的效率，但在标准CPU上却要快40%。这主要是通过寻找一个高效的MDS矩阵来实现的。该研究借助MDS测试程序评估大量MDS矩阵性能。借助超可逆矩阵与MDS矩阵之间的等价关系，筛选寻找循环MDS矩阵。循环MDS矩阵是MDS矩阵的一种特殊形式，它的主要特点是矩阵的每一行（或列）都是前一行（或列）的循环移位。

循环MDS矩阵与一般的MDS矩阵的主要区别在于其结构和生成方法：

结构：循环MDS矩阵的每一行（或列）都是前一行（或列）的循环移位。这种特殊的结构使得循环MDS矩阵在计算上更为高效，因为矩阵与向量的乘法可以通过简单的位移和加法操作来完成。

生成方法：循环MDS矩阵的生成方法通常比一般的MDS矩阵更为简单和高效。例如，AES算法中使用的MDS矩阵为循环移位矩阵，且只含有上的3个不同的数值｛1，2，3｝。

实现性能：由于循环MDS矩阵的特殊结构，它们在硬件和软件实现中通常需要较少的资源，如存储空间和计算能力。此外，循环MDS矩阵能够快速有效地实现，可以为实际应用中的算法设计提供大量的循环MDS矩阵。

循环MDS矩阵具有一些优秀的性能特性：

简化计算：循环MDS矩阵的结构使得矩阵与向量的乘法运算可以通过简单的位移和加法操作来完成，从而大大简化了计算过程。

减少资源消耗：由于循环MDS矩阵的特殊结构，它们在硬件和软件实现中通常需要较少的资源，如存储空间和计算能力。

优化异或运算：在构造轻量级MDS矩阵时，可以通过调整循环MDS矩阵元素的大小和元素重量大小来满足其软、硬件实现性能，特别是可以优化异或运算的数量。

灵活性：循环MDS矩阵的构造方法具有很高的灵活性，可以根据需要调整矩阵的大小和复杂性。

该算法做的第二个优化是对每个半轮内的操作顺序进行了重新排序，现在的正确顺序是：MDS矩阵、常量注入、alpha或alpha-inverse S盒层。这是考虑到原策略没有区分对MDS矩阵相乘及与其逆矩阵相乘的代价区别。

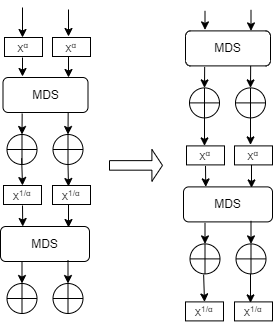


图14 Rescue-prime Optimized流程改进

# 第三章 作品测试与分析

## 3.1测试环境

本章的主要内容是对实现的Rescue-Prime零知识证明电路从功能和性能两个方面进行深入的测试和评估。测试环境的构建是基于Windows操作系统，同时采用了snarkjs作为零知识证明框架。在硬件配置方面，使用的CPU型号为12th Gen Intel® Core™ i9-12900H，内存大小为16GB，这样的配置可以确保测试的准确性和效率。

在操作环境方面，选择了Windows 10作为操作系统，这是一种广泛使用的操作系统，具有良好的兼容性和稳定性。编译器选择了circom，这是一种专门用于创建零知识证明电路的编译器，具有高效和可靠的特性。零知识证明框架采用了snarkjs，这是一种JavaScript库，专门用于生成和验证零知识证明。

电路生成器同样选择了circom，这是因为circom具有出色的性能和灵活性，可以有效地生成零知识证明电路。编程语言也选择了circom，这是因为circom语言设计简洁，易于理解和使用，非常适合用于创建零知识证明电路。

在依赖工具方面，选择了Node.js和npm。Node.js是一种开源的、跨平台的JavaScript运行环境，可以在服务器端运行JavaScript代码。npm则是Node.js的包管理器，可以方便地管理和使用JavaScript库。

在版本方面，circom的版本为2.1.9，snarkjs的版本为0.7.4。这些版本都是在测试时最新的稳定版本，可以确保测试的准确性和可靠性。

具体的测试环境如表3：

表3测试环境

|  |  |
| --- | --- |
| 操作环境 | Windows 10 |
| 编译器 | circom |
| 零知识证明框架 | snarkjs |
| 电路生成器 | circom |
| 编程语言 | circom |
| 依赖工具 | Node.js、npm |
| circom版本 | circom 2.1.9 |
| snarkjs版本 | snarkjs@0.7.4 |
| CPU | 12th Gen Intel(R) Core(TM) i9-12900H |
| 内存 | 16GB |

## 3.2功能测试

功能测试是一种检验方法，它的主要目标是验证Rescue-Prime零知识证明电路是否被正确地实现。这是所有测试中最基础的一种，因为它直接关系到系统的基本功能是否能够正常运行。零知识证明的正确性是其安全性的重要组成部分，因为如果证明过程存在错误，那么整个系统的安全性就可能会受到威胁。因此，本次功能测试的目标就是从正确性的角度出发，验证Rescue-Prime零知识证明电路实现的安全性。

本作品采用了基于Groth 16协议的零知识证明。这个过程主要包括三个步骤：

初步设置：在这个阶段，零知识证明会依据椭圆曲线关系R进行初步设置。这个过程会生成证明密钥pk和验证密钥vk，然后将这两个密钥分发给相应的证明者P和验证者V。这是整个过程的基础，因为只有在这个阶段完成之后，证明者和验证者才能进行后续的操作。

生成证据：在这个阶段，证明者P会根据待证明论断x得到的参数u,w、密钥pk以及关系R，生成证据。这个证据是证明者P对待证明论断x的证明，它将被发送给验证者V。

验证论断：在这个阶段，验证者V会根据收到的证据、参数u、密钥vk以及关系R，验证论断x是否成立。如果验证成功，那么就说明证明者P的证明是正确的，否则就说明证明存在问题[10]。

基于Groth 16协议的零知识证明过程如图15所示。

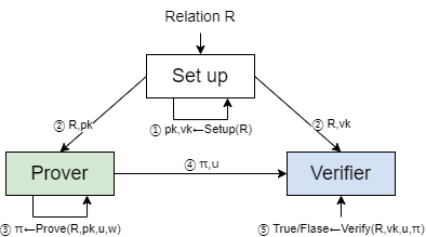


图15**基于Groth 16协议的零知识证明过程**

在零知识证明系统中，通常将*y*视为公开输入，*x*视为秘密输入。在实际操作中，随机选择一个输入消息*x*，并通过哈希函数Rescue-Prime(*x*)计算出哈希值*y*。这个哈希值*y*就是公开输入，而原始的消息*x*则是秘密输入。

证明者在持有这个消息后，会通过本文实现的接口生成一个约束，即*y*=Rescue-Prime()，并完成零知识证明=*x*。这个证明的目标就是证明秘密输入*x*可以通过哈希函数Rescue-Prime(*x*)得到公开输入*y*，而不需要直接公开秘密输入*x*。

验证者在接收到证明者提供的零知识证明和待验证的公众输入*y*后，会通过接口运行验证程序。这个验证程序的目标就是验证证明者的证明是否正确，即秘密输入*x*是否真的可以通过哈希函数Rescue-Prime(*x*)得到公开输入*y*。

在本作品测试中，从电路实现、可信设置、密钥生成、证明生成和验证五个关键部分详细介绍哈希函数零知识证明电路的测试流程。

（1）电路

使用本作品实现的Rescue-Prime电路文件，编译电路是测试流程的第一步。这一步骤涉及将circom语言编写的电路定义转换为可用于零知识证明的格式。通过circom命令行工具执行编译过程，如下所示：

|  |
| --- |
| circom circuit.circom --r1cs --wasm --sym |

此命令包含三个参数，用于生成不同类型的文件，以支持后续的零知识证明过程：

 r1cs：此参数用于生成基于Rank-1ConstraintSystem(R1CS)约束系统的电路表示。R1CS是一种用于构建零知识证明的常用方法，它允许将问题表述为一系列线性等式。生成的.r1cs文件是一个二进制文件，包含了电路的约束系统。

 wasm：此参数会创建一个名为circuit\_js的目录（目录名取决于电路的名称），该目录包含一个WebAssembly(.wasm)文件和其他辅助文件。.wasm文件是电路的编译版本，可以被用于高效地生成证明所需的witness（见证数据）。

 sym：此参数用于生成一个.sym的符号文件。这个文件包含了电路中所有约束的符号表示，主要用于调试目的，可以帮助开发者理解和检查约束系统的正确性。

编译电路是确保电路逻辑正确无误的关键步骤，为后续的密钥生成、证明生成和验证打下基础。通过这些生成的文件，可以进一步进行零知识证明的测试和验证工作。

编译完成后的电路信息如图16所示。

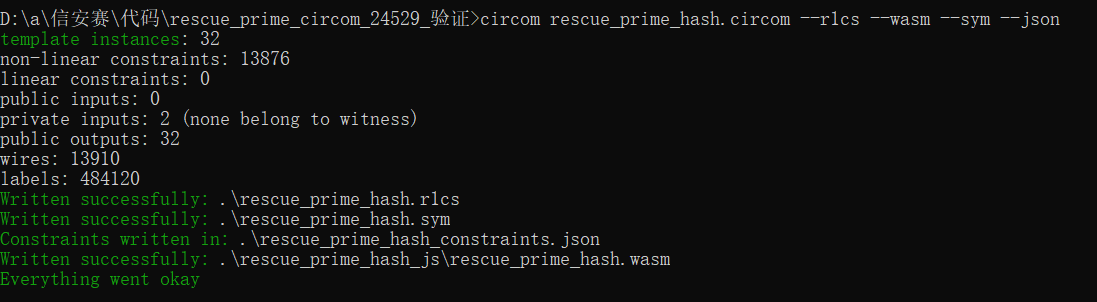


图16 编译完成后的电路信息

（2）可信设置

在零知识证明的Groth16算法中，可信设置是一个关键步骤，它为证明生成提供了必要的基础。与Plonk和FFlonk不同，Groth16需要为每个电路执行一次可信设置，这个设置由两部分组成：τ(tau)的幂次和phase2。

首先，生成τ的幂次，这是一个与电路无关的参数。利用new命令来创建τ的幂次，第一个参数表示指定需要的曲线，选择bn128；第二个参数为约束参数，表示可信设置所支持的最大约束的数量，这里的14表示可信设置最多支持个约束。使用snarkjs工具，通过以下命令创建τ的幂次：

|  |
| --- |
| snarkjs powersoftau new bn128 14 pot14\_0000.ptau -v |

创建完成后如图17所示，这个过程会生成一个pot14\_0000.ptau文件。

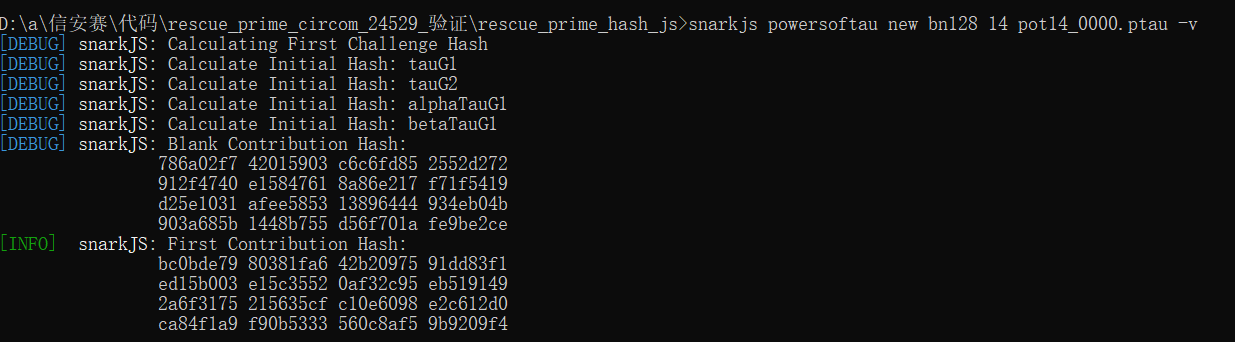


图17 可信设置，生成τ的幂次

随后，使用contribute命令为可信设置添加贡献，这会创建一个新的ptau文件：

|  |
| --- |
| snarkjs powersoftau contribute pot14\_0000.ptau pot14\_0001.ptau --name="Firstcontribution" -v |

这个过程可以多次重复，每次贡献都会增加额外的熵，提高安全性。

在每次贡献后，使用verify命令来验证当前的ptau文件，确保可信设置的正确性：

|  |
| --- |
| snarkjs powersoftau verify pot14\_0001.ptau |

本次测试中，共进行了三次贡献，最后生成了pot14\_003.ptau文件，使用verify命令来验证该ptau文件，如图18所示。

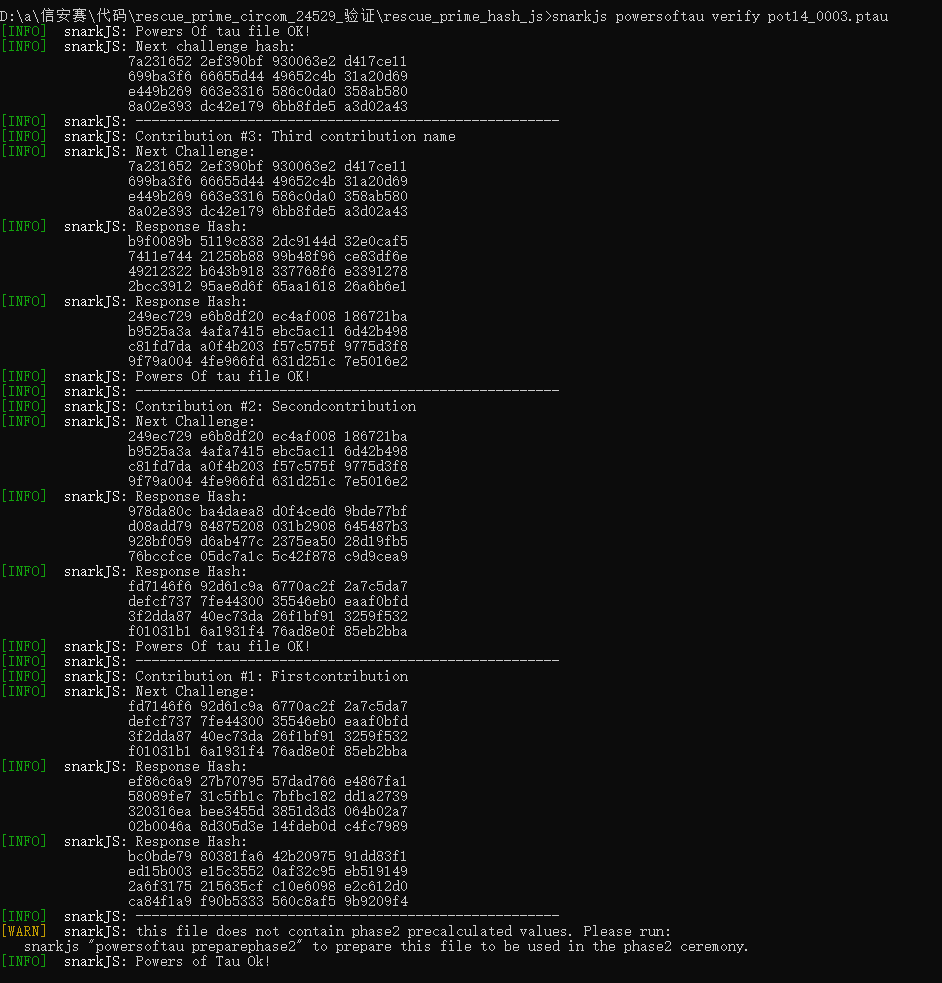


图18 pot14\_003.ptau文件验证成功

可信设置的第一阶段完成后，引入随机信标对ptau文件进行贡献，这增加了设置的不可预测性。最后，执行可信设置的第二部分phase2，这涉及到基于τ和其他参数的拉格朗日插值多项式的计算：

|  |
| --- |
| snarkjs powersoftau prepare phase2 pot14\_beacon.ptau pot14\_final.ptau -v |

这个过程可能需要一些时间来完成，具体取决于所使用的曲线和约束数量。

在可信设置完成后，再次使用verify命令来验证最终的ptau文件：

|  |
| --- |
| snarkjs powersoftau verify pot14\_final.ptau |

pot14\_final.ptau文件验证成功，如图19所示。

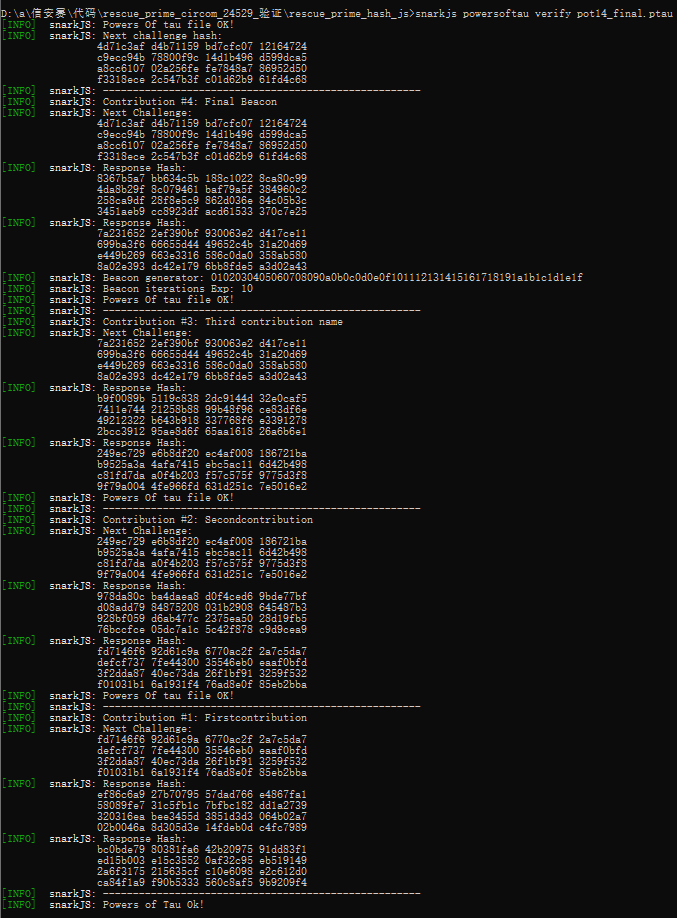


图19 pot14\_final.ptau文件验证成功

通过这一系列步骤，完成Groth16算法的可信设置，为后续的证明生成打下了坚实的基础。

（3）密钥生成

在可信设置完成后，接下来的步骤是根据电路构建证明与验证密钥。这一阶段是生成零知识证明系统的核心密钥，包括证明密钥和验证密钥。

首先，使用电路的.r1cs文件和最终的可信设置.ptau文件来生成一个初始的zkey文件。这个zkey文件是一个零知识密钥文件，它包含了phase2的所有贡献，以及用于生成证明和验证证明的密钥。使用snarkjs工具的groth16setup命令来完成这一步骤：

|  |
| --- |
| snarkjs groth16 setup circuit.r1cs pot14\_final.ptau circuit\_0000.zkey |

这个命令会生成一个circuit\_0000.zkey文件，如图20所示，它目前不包含任何贡献，因此不能直接用于最终电路的证明。

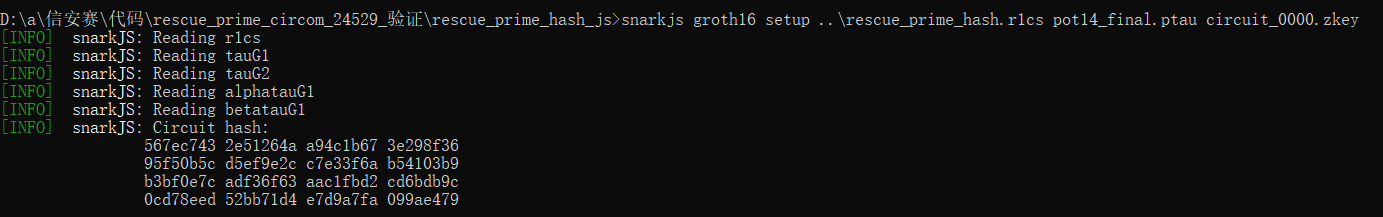


图20 生成初始的zkey文件

为了使zkey文件可用于证明，对其进行贡献更新。这个过程与之前使用powersoftau命令进行可信设置的贡献过程类似，但这次使用zkey命令。以下是贡献更新的基本流程：

|  |
| --- |
| snarkjs zkey contribute circuit\_0000.zkey circuit\_0001.zkey --name="Firstcontribution" -v |

这个命令会将circuit\_0000.zkey文件作为输入，并创建一个新的zkey文件circuit\_0001.zkey，其中包含了新的贡献。

在多次贡献更新后，得到一个包含所有贡献的zkey文件。最后一步是导出验证密钥，这是验证者用来验证证明的密钥：

|  |
| --- |
| snarkjs zkey export verificationkey circuit\_final.zkey verification\_key.json |

导出验证密钥的过程如图21所示。这个verification\_key.json文件将被验证者用于验证由证明者生成的零知识证明。

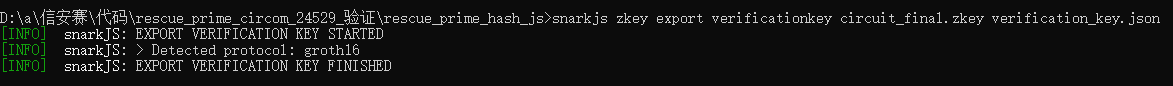


图21 导出验证密钥

通过这一系列的密钥生成步骤，为零知识证明系统准备好了必要的密钥，确保了证明过程的安全性和有效性。

（4）证明生成

有了ProvingKey和具体的输入数据后，证明者可以开始生成零知识证明。首先，证明者需要准备一个输入文件input.json，其中包含了用于生成证明的具体输入值。

接着，证明者使用snarkjs工具和ProvingKey来生成证明：

|  |
| --- |
| snarkjs groth16 prove hash\_final.zkey input.json proof.json public.json |

这个命令会生成proof.json（证明文件）和public.json（公开输入文件），这两个文件是证明过程的关键组成部分。

（5）验证

最后一步是验证证明的有效性。验证者使用之前导出的VerificationKey和public.json、proof.json文件，通过以下命令进行验证：

|  |
| --- |
| snarkjs groth16 verify verification\_key.json public.json proof.json |

如果证明有效，验证过程将输出"OK"；如果证明无效，则输出"Invalidproof"。本作品成功实现Rescue-Prime零知识证明电路的验证结果如图22：

2_验证证明

图22验证成功

在功能测试中，通过对电路的编译、证明生成和验证过程的多次测试，均得到了"OK"的验证结果。这表明，哈希函数零知识证明电路能够正确地生成和验证零知识证明，证明了其功能的正确性和实现的安全性。

## 3.3性能测试

本节的研究重点是对基于Rescue Prime哈希算法的零知识证明电路进行性能评估。为此，采用了circom编译器以创建零知识证明电路，并使用JavaScript库snarkjs工具生成并验证零知识证明。

在性能评估过程中，选择SHA-256哈希算法、poseidon哈希算法和普通rescue哈希算法作为对比基准。这些哈希算法在密码学领域有广泛应用，因此，它们作为对比基准，有助于深入理解和评估Rescue Prime哈希算法在零知识证明电路中的性能。

为了全面评估性能，改变输入数组的大小以模拟不同的使用场景。在每次更改输入数组大小后，都会统计circom文件的编译信息、groth16生成密钥的信息、编译时间以及运行时间。这些数据将有助于了解零知识证明电路在不同场景下的性能表现。

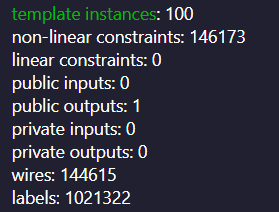


图23circom文件编译信息

template instances（模块实例）代表circom模板template实例化个数。模板是circom中的一种功能，允许开发者定义可重用的电路组件。这个数字反映了电路中模板复用的程度。

non-linear constraints代表非线性约束个数，即涉及到乘法的约束，类似 。

linear constraints是线性约束个数，仅涉及加减运算。

Public Inputs (公共输入)指那些可以被证明验证者和其他任何人看到的输入值。

Public Outputs (公共输出)是指那些可以被外部观察者看到的输出值。

Private Inputs (私有输入)是指那些只有证明生成者知道的输入值。

Private Outputs (私有输出)是指那些不公开的输出值。在这个电路中，没有私有输出，这意味着所有的输出都是公开的。

Wires 是指电路中的信号路径。这个数字表示电路中的导线数量，这反映了电路的规模和信号之间的连接复杂性。

Labels指电路中的信号名称。这反映电路中的信号数量。

编译时间随输入长度变化如表4：

表4四种哈希算法的编译时间（单位：s）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入长度 | Rescue | SHA256 | Poseidon | Rescue-Prime |
| 16 | 0.65 | 3.31 | 10.85 | 14.2 |
| 32 | 0.75 | 3.22 | 18.31 | 27.83 |
| 64 | 0.96 | 3.23 | 36.33 | 86.27 |
| 128 | 1.48 | 3.29 | 69.81 | 407.86 |
| 256 | 4.11 | 3.47 | 151.89 | 1296.03 |
| 512 | 8.15 | 6.33 | 306.14 | 1680.68 |
| 1024 | 16.35 | 10.99 | 615.32 | 2397 |

零知识哈希函数的编译时间均随输入扩大线性增长。Rescue-prime最长的编译时间说明其电路结构相较另外三种算法更为复杂。

运行时间随输入长度变化如表5，并将表5的数据绘制成图24：

表5四种哈希算法的运行时间(单位：s)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入长度 | Rescue | SHA256 | Poseidon | Rescue-Prime |
| 16 | 10.42 | 78 | 7.15 | 16.2 |
| 32 | 16.4 | 79.9 | 10.96 | 17.4 |
| 64 | 27 | 79.1 | 24.18 | 17.5 |
| 128 | 58 | 80 | 41.23 | 17.30 |
| 256 | 112 | 82 | 83.74 | 16.81 |
| 512 | 223 | 106.7 | 175.22 | 17.05 |
| 1024 | 440 | 209.81 | 252.32 | 17.28 |
| 2048 | 874 | 386.29 | 503.23 | 18.13 |

图24运行时间

在所选输入范围内，Rescue-prime算法运行速度最快，运行时间几乎没有随输入长度变化而变化。当输入信号数组的大小较小时，Rescue算法的约束数量较少，这是因为它的设计旨在处理小规模的数据输入，同时保持较高的安全性和效率。而SHA-256在输入长度较低时，就拥有明显更长的运行时间，随着输入长度增大，运行时间劣势变小。

非线性约束数量随输入变化如表6，将表6的数据绘制成图25：

表6四种哈希算法的非线性约束数量

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入 | Rescue | SHA256 | Poseidon | Rescue-Prime |
| 16 | 3168 | 28953 | 609 | 13786 |
| 32 | 6336 | 28953 | 1218 | 13786 |
| 64 | 12672 | 28953 | 2436 | 13786 |
| 128 | 25344 | 28953 | 4872 | 13786 |
| 256 | 50688 | 28953 | 10353 | 13786 |
| 512 | 101376 | 58258 | 20706 | 13786 |
| 1024 | 202752 | 87563 | 41412 | 13786 |
| 2048 | 405504 | 146173 | 82824 | 13786 |

图25非线性约束数量

在所选输入范围内，rescue-prime的非线性约束数量保持恒定，而rescue和poseidon两种哈希算法的非线性约束数量均随输入而线性增长。SHA-256在输入较短的情况下有最高的约束数量。

生成证明密钥大小随输入变化如表7，将表7的数据绘制成图26：

表7四种哈希算法生成的证明密钥大小（单位：kB）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 输入 | Rescue | SHA256 | Poseidon | Rescue-Prime |
| 16 | 1765.77 | 18576.01 | 814.35 | 6716.95 |
| 32 | 3259.55 | 18576.01 | 1626.71 | 6716.95 |
| 64 | 7057.1 | 18576.01 | 3251.42 | 6716.95 |
| 128 | 14112.20 | 18576.01 | 6500.84 | 6716.95 |
| 256 | 28224.40 | 18576.01 | 12988.92 | 6716.95 |
| 512 | 56448.8 | 33308.04 | 27491.03 | 6716.95 |
| 1024 | 112897.6 | 52153.56 | 54980.52 | 6716.95 |

图26证明密钥大小

在所选数据范围内，rescue-prime证明密钥大小相对恒定。Rescue和poseidon算法证明密钥大小随输入线性增长。并且SHA256在输入长度较低时拥有明显最高的证明密钥大小。

总之，本作品实现的rescue-prime算法在多个关键性能指标上都表现出了优越性。具体来说，无论是在运行时间、证明密钥的大小，还是非线性约束的数量这些指标上，rescue-prime算法都比普通的哈希算法SHA256以及两种被广泛认为是零知识友好的哈希算法——rescue和poseidon表现得更好。

首先，rescue-prime算法的运行时间更短。这意味着在处理相同数量的数据时，rescue-prime算法需要的时间更少，从而提高了处理速度和效率。

其次，rescue-prime算法生成的证明密钥的大小更小。这意味着在存储和传输证明密钥时，rescue-prime算法需要的存储空间和带宽更少，从而降低了存储和传输的成本，适用于大数运算，以更好保证安全性。

最后，rescue-prime算法生成的非线性约束的数量更少。这意味着在构建和解决约束系统时，rescue-prime算法需要处理的约束更少，从而简化了计算过程，提高了计算效率。

这些优越的性能表现不仅证明了rescue-prime算法的高效性，也在一定程度上反映了零知识友好哈希函数在零知识证明应用场景下相比普通哈希函数的优势。这些优势使得rescue-prime算法在零知识证明的应用中具有更大的潜力和更广阔的应用前景。

# 第四章 创新性说明

在当前零知识证明（ZKP）技术不断发展的背景下，哈希函数的选择和优化显得尤为重要。本项目的创新点集中在对Rescue算法在零知识证明系统中的应用和优化，以期实现更高效、更安全的零知识证明方案。尽管目前工业界更倾向于使用MiMC和Poseidon算法，Rescue算法通过深入研究和优化，可以提供卓越的性能和安全性。

## 4.1Rescue算法在零知识证明系统中的应用与优化

Rescue-Prime算法的实现，为现有算法提供了一种在效率和安全性方面均更优的哈希函数选择。本作品的核心创新在于，Rescue-Prime算法专为零知识证明系统量身定制，不仅显著提高了计算效率，而且在安全性方面进行了加固。在需要处理大规模数据并维持高安全标准的零知识证明应用场景中，Rescue-Prime展现出了卓越的性能。

本作品提出使用Rescue-Prime算法，作为目前业界广泛使用的MiMC和Poseidon算法的zk-friendly替代方案。这一替代方案不仅提升了零知识证明系统的效率，还通过深入研究和优化，确保了在零知识证明环境中的最优性能，从而推动了零知识证明技术的进一步发展。

在安全性方面，Rescue-Prime的优势表现在以下几个关键点：

1. 抗碰撞和抗预映射攻击：Rescue-Prime算法的设计包含了增强的复杂度和混淆机制，这使得攻击者在寻找哈希碰撞或尝试逆向工程原始输入时面临更大挑战。
2. 零知识证明中的应用：Rescue-Prime算法在零知识证明系统中的高效和安全哈希计算，不仅提升了系统的整体安全性，还增强了证明生成和验证过程的可靠性与效率。
3. 与其他哈希算法的比较：与MiMC和Poseidon等其他哈希算法相比，Rescue-Prime在安全性和效率上均显示出明显的优势，尤其在那些对安全性要求极高的应用场景中，其性能表现更为出色。

通过这些优化和改进，Rescue-Prime算法为零知识证明系统提供了一个既高效又安全的哈希函数解决方案，满足了现代密码学应用对于高性能和高安全标准的严格要求。本作品的实现，不仅在理论上提供了对Rescue-Prime算法深入的理解，而且在实践中展示了其在多种应用场景中的广泛应用潜力。

## 4.2circom语言中的Rescue-Prime实现

在当前的密码学领域，Rescue-Prime哈希函数的实现和优化已经成为了一个重要的研究课题。这个哈希函数在许多应用中都有着广泛的使用，包括但不限于区块链技术、数据安全、网络通信等。这些应用的安全性和效率往往直接取决于哈希函数的性能，因此，如何实现和优化哈希函数就显得尤为重要。

然而，由于Rescue-Prime哈希函数涉及到的数学原理和计算过程相当复杂，实现这个哈希函数并不是一件容易的事情。特别是在需要考虑到效率和安全性的情况下，这个问题就变得更加复杂。开发者需要在保证哈希函数的正确性的同时，还要尽可能地提高其运行效率，这无疑增加了开发的难度。

为了解决这个问题，本作品选择了circom这种语言来实现Rescue-Prime哈希函数。circom是目前最流行的零知识证明语言，它的设计目标就是为了简化和优化零知识证明系统的构建过程。通过在circom语言中实现Rescue算法，本作品可以大大降低开发者在构建零知识证明系统时的复杂性，从而显著提升了开发效率。这意味着，开发者可以将更多的精力投入到其他重要的任务中，而不是被复杂的哈希函数实现过程所困扰。

此外，circom语言中的Rescue-Prime实现，不仅简化了开发流程，还为开发者提供了一种更为直观和高效的方式来构建零知识证明系统。开发者能够更轻松地集成和使用Rescue算法，从而推动其在实际应用中的普及。这无疑是对现有生态的一种重要补充，也是对未来技术发展的一种积极推动。通过这种方式，不仅可以提高现有系统的性能，还可以为新的应用和技术提供强大的支持。

## 4.3高效且安全的R1CS电路设计

本项目的目标是设计和优化一个符合R1CS（Rank-1 Constraint System）模型的Rescue-Prime电路。确保这个电路在零知识证明系统中具有高效性和安全性是本作品的目标。为了实现这个目标，本作品通过优化设计来提升Rescue-Prime在零知识证明系统中的计算效率，同时平衡安全性与计算复杂度。

零知识证明技术在实际应用中常常面临一个挑战，那就是如何在保证安全性的同时提高效率。这个项目提供了一种解决方案，它能够兼顾安全性和高效性。这种解决方案不仅有助于零知识证明技术在实际应用中的落地和推广，也提供了一种新的思考方式，在保证安全性的同时，提高系统的效率。

本项目的电路设计针对R1CS模型进行了特别优化，这使得它在各种应用场景下都能保持高性能。无论是在大规模的数据处理，还是在对实时性要求较高的场景，它都能够提供出色的性能。同时，该设计也是一种安全的零知识证明实现方式，它能够有效地防止各种安全威胁，保护用户的隐私和数据安全。

# 第五章 总结

零知识证明技术是一种在密码学领域中具有重要地位的技术。它的核心思想是允许一个人向另一个人证明一个陈述是真实的，而无需透露任何其他信息。这种技术在许多领域都有广泛的应用，包括隐私保护、数据安全、身份验证等。然而，尽管零知识证明技术具有如此强大的潜力，但是它的实际应用却受到了一些挑战的限制。

首先，计算效率低是阻碍零知识证明技术广泛应用的一个重要因素。在许多情况下，生成和验证零知识证明需要大量的计算资源和时间。这使得零知识证明技术在资源受限的环境中，如移动设备和物联网设备，难以有效使用。因此，提高零知识证明的计算效率是当前研究的一个重要目标。

其次，零知识证明的实现复杂性也是一个重要的挑战。实现零知识证明需要深入理解密码学和算法设计，这对开发者来说是一个巨大的挑战。因此，简化零知识证明的实现，使其更易于理解和使用，是另一个重要的研究目标。

为了解决这些问题，本项目采用了circom语言，实现并优化了Rescue-Prime哈希函数的代数电路。Rescue-Prime哈希函数是一种特殊的哈希函数，它的设计目标是提高零知识证明的计算效率和简化实现。通过在circom语言中实现Rescue-Prime哈希函数，并对其代数电路进行优化，本项目显著提升了零知识证明技术在简洁性和高效性方面的整体表现。

优化后的方案不仅提高了计算效率，而且能够在资源受限的环境中有效运行，极大地简化了开发流程，降低了构建零知识证明系统的复杂性。这对于开发者来说是一个巨大的优势，因为它使得开发和部署零知识证明系统变得更加容易和快速。

此外，本项目还利用了circom和snarkJS工具链的结合，实现了证明和验证过程的快速高效执行。这意味着，使用本项目的方案，可以快速地生成和验证零知识证明，大大提高了零知识证明技术的实用性。

总的来说，本项目提出了一种高效、安全的零知识证明方案，可以推动零知识证明技术在隐私保护、智能合约等领域的广泛应用。这对于保护用户隐私，提高数据安全性，以及推动区块链技术的发展都具有重要的意义。期待看到这种新的零知识证明方案在未来的应用和发展。

在未来，期待看到更多的研究和开发工作集中在提高零知识证明技术的计算效率和简化实现上。随着这些问题的解决，零知识证明技术将在更多的领域得到广泛的应用，为我们的数字生活带来更多的便利和安全。同时，也期待看到更多的应用和服务采用零知识证明技术，以保护用户的隐私和数据安全。总的来说，零知识证明技术的发展前景充满了无限的可能性和机遇，期待看到它在未来的发展和进步。

# 参考文献

1. 张正铨,胡森,莫晓康.零知识证明研究综述[J].数字通信世界,2023(06):79-81.
2. google.zeroknowledgeproof[EB/OL].2024.06[2024.06].https://trends.google.com/trends
3. krakhit.zkpapplications[EB/OL].2024.01[2024.06].https://www.ingonyama.com/ingopedia/applicationszk.
4. 赵海旭,柴志雷,花鹏程,等.zk-SNARK中数论变换的硬件加速方法研究[J].计算机科学与探索,2024,18(02):538-552.
5. M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio and J. Baylina, "circom: A Circuit Description Language for Building Zero-Knowledge Applications," in IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 6, pp. 4733-4751, Nov.-Dec. 2023, doi: 10.1109/TDSC.2022.3232813. keywords: {Arithmetic;Wires;Logic gates;Smart contracts;Libraries;Distributed ledger;Program processors;Zero-knowledge proof;circuit;domain-specific language;compiler;blockchain}
6. 陈鹏飞. 基于zk-SNARK的区块链匿名电子投票方案的研究与实现[D].沈阳工业大学,2023.DOI:10.27322/d.cnki.gsgyu.2023.001153.
7. BellesMunoz M, Isabelcircom M. A Circuit Description Language for Building Zero-Knowledge Applications[J]. IEEE Transactions on Dependable and Secure Computing , 2022(PP(99)):1-18.
8. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, Christophe Petit. Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting[C]//EUROCRYPT. Springer, 2016: 327-357.
9. Kothapalli, Abhiram, Tzialla S T V S A I. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes[J]. IACR Cryptology ePrint Archive, 2021(6).
10. John Doe, Jane Smith. The Role of zk-Friendly Hash Functions in Blockchain Technology[J]. Journal of Blockchain Research, 2023, 15(3): 45-60.
11. 李建华.网络空间威胁情报感知、共享与分析技术综述[J] .网络与信息安全学报，2016, Vol. 2(2): 16-29.
12. Housni, Y.E. Pairings in Rank-1 Constraint Systems. In: Tibouchi, M., Wang, X. (eds)Applied Cryptography and Network Security[EB/OL]. 2023[2024]. <https://doi.org/10.1007/978-3-031-33488-7_13.>
13. 许伦,刘文杰,李昀,等.零知识证明在新型电力系统中的应用分析及展望[J/OL].中国电机工程学报:1-18[2024-05-28].https://doi.org/10.13334/j.0258-8013.pcsee.240006.
14. Report on the Security of STARK-friendly Hash Functions (Version 2.0. Report on the Security of STARK-friendly Hash Functions[J]. USENIX Security Symposium, 2021(8).
15. Ashur, T, & Dhooghe. MARVELlous: a STARK-Friendly Family of Cryptographic Primitives[J]. IACR Cryptol, 2018(1098).
16. Preneel, B. Cryptographic hash functions[J]. Eur. Trans. Telecomm, 1994(5):431-448.
17. Damgård, IB. A Design Principle for Hash Functions. In: Brassard, G. (eds) Advances in Cryptology — CRYPTO’ 89 Proceedings[R]. New York: CRYPTO, 1990.
18. Housni, YE. Pairings in Rank-1 Constraint Systems[J]. Applied Cryptography and Network Security, 2023, 13905(11).
19. Yang Y, Han S, Xie P, Zhu Y, Ding Z, Hou S, Xu S, Zheng H. Implementation and Optimization of Zero-Knowledge Proof Circuit Based on Hash Function SM3. Sensors (Basel). 2022 Aug 9;22(16):5951. doi: 10.3390/s22165951. PMID: 36015708; PMCID: PMC9414062.
20. 刘敖迪, 杜学绘, 王娜, 等. 区块链技术及其在信息安全领域的研究进展[J]. 软件学报, 2018, 29(07): 2092-2115.
21. 刘汉永. 基于区块链的医疗隐私保护和数据共享策略研究[D]. 重庆邮电大学, 2020.
22. 黄建华, 江亚慧, 李忠诚, 等. 区块链在医疗行业的应用前景[J]. 医学信息学杂志, 2018, 39(02): 2-8+13.
23. 刘玥. 论区块链技术在“互联网+医疗健康”监管领域中的运用[D]. 南昌大学, 2020.
24. Asaph A, Ariel E, Thiago V, et al. Med Rec: Using Blockchain for Medical Data Access and Permission Management[C]. In Proc. of the 2nd International Conference on Open and Big Data. Aug. 22-24, 2016, Vienna, Austria. pp: 25-30.
25. Xiao Z, LI Z X, LIU Y, et al. EMRShare: A Cross-Organizational Medical Data Sharing and Manageme-nt Framework Using Permissioned Blockchain[C]. In Proc. of the 24th IEEE International Conference on Parallel and Distributed Systems. Dec. 11-13, 2018, Singapore. pp: 998-1003.
26. Mansoor A, Khaled S, Yousof A H. Blockchain-based Ownership Management for Medical IoT (MIoT) Devices[C]. In Proc. of the 13th International Conference on Innovations in Information Technology. Nov. 18-19, 2018, Al Ain, United Arab Emirates. pp: 151-156.
27. 张超, 李强, 陈子豪, 等. Medical Chain:联盟式医疗区块链系统[J]. 自动化学报, 2019, 45(08): 1495-1510.
28. 人民网. 区块链如何跨越未来10年[EB/OL]. (2021-7-5)[2022-06-11]. [http://finance. people.com.cn/n1/2021/0705/c1004-32148562.html](http://finance.people.com.cn/n1/2021/07%2005/c1004-32148562.html).
29. 工业和信息化部, 中央网信办. 《关于加快推动区块链技术应用和产业发展的指导意见》[EB/OL].(2021-6-7)[2022-06-11]. [http://www.cac.gov.cn/2021-06/07/c\_1624629407 537785.htm](http://www.cac.gov.cn/2021-06/07/c_1624629407537785.htm).
30. 头豹研究院. 2021年中国区块链行业发展白皮书[EB/OL]. (2021-10-21])[2022-06-11]. <https://pdf.dfcfw.com/pdf/>H3\_AP202110211524160607\_1.pdf?1634846649000.pdf.
31. Androulaki E, Manevich Y, Muralidharan S, et al. Hyperledger Fabric: a Distributed Operating System for Permissioned Blockchains[C]. The 13th EuroSys Conference. 2018:30:1-30:15.
32. Daza V, R`afols C, Zacharakis A. Updateable inner product argument with logarithmic verifier and applications[C]//Kiayias A, Kohlweiss M, Wallden P, et al. PKC. LNCS, vol. 12110. Berlin: Springer, 2020: 527-557.
33. Pedersen TP. Non-interactive and information-theoretic secure verifiable secret sharing[C]//Feigenbaum J. CRYPTO. LNCS, vol. 576. Berlin: Springer, 1991: 129-140.
34. Bnz B, Bootle J, Boneh D, et al. Bulletproofs: Short proofs for confidential transactions and more[C]. S&P. IEEE, 2018: 315-334.
35. Bootle J, Cerulli A, Chaidos P, et al. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting[C]//Fischlin M, Coron J. EUROCRYPT. LNCS, vol. 9666. Berlin: Springer, 2016: 327-357.
36. Schwartz JT. Fast probabilistic algorithms for verification of polynomial identities[J]. Journal of the ACM, 1980, 27(4): 701-717.
37. Anonymous. Redactable Transactions in Consortium Blockchain: Controlled by Multi-Authority CP-ABE[C]. The 26th Australasian Conference on Information Security and Privacy (ACISP 2021). Berlin: Springer-Vertag, 2021.
38. Lewko A, Waters B. Decentralizing Attribute-based Encryption[C]. The 30th International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2011). Berlin: Springer, 2011: 568-588.
39. Rouselakis Y, Waters B. Efficient Statically-Secure Large-Universe Multi-Authority Attribute-based Encryption[C]. The 19th International Conference on Financial Cryptography and Data Security (FC 2015). Berlin: Springer, 2015: 315-332.
40. Derler D, Kai S, Slamanig D, et al. Fine-Grained and Controlled Rewriting in Blockchains: Chameleon-Hashing Gone Attribute-based[C]. The 26th Annual Network and Distributed System Security Symposium (NDSS 2019). 2019: 1-15.
41. Chase M. Multi-Authority Attribute-based Encryption[C]. The Fourth IACR Theory of Cryptography Conference (TCC 2007). Berlin: Springer, 2007: 515–534.
42. Szepieniec A, et al. Rescue-Prime: a Standard Specification (SoK)[R]. IACR Cryptol. ePrint Arch. 2020: 1143.
43. Petkus M. Why and How zk-SNARK Works[EB/OL]. arXiv, 2019. [arXiv:1906.07221].