摘 要

2008 年，中本聪提出了比特币的概念，随后比特币问世，逐渐走入公众视野。随着比特币技术的成熟和应用，数字资产管理面临着越来越严峻的安全挑战。为解决这一问题，MPC 钱包作为一种新型解决方案应运而生，通过私钥拆分和分散管理，提高了数字资产的安全性。然而，MPC 钱包的底层技术——ECDSA 门限签名技术在保障安全性的同时存在运算成本高和隐私性差的问题。为解决该问题，本文进行了以下研究工作：

1、在当前门限签名方案存在隐私性不足的情况下，本研究引入了用户匿名性的概念。用户匿名性旨在确保多个节点服务多个用户时所生成的中间数据无法被关联到特定用户。该特性有助于防止多个节点串通，以获取有关用户的敏感信息。通过情景分析，我们总结了实现用户匿名性的条件，并在后续的方案设计中满足了这些条件。

2、提出了一种名为 TB-ECDSA 的签名方案。针对当前 ECDSA 门限签名方案运算成本高和隐私性差的问题，通过结合门限签名和盲签名，我们提出了 TB-ECDSA 方案。该方案利用同态加密技术，使得签名者无法从签名的过程中提取有用的信息，从而保证了用户的隐私性。同时，方案具有用户匿名性，签名者之间无法将同一个用户的数据关联起来，进一步保证了用户的隐私性。并且，经过实际测试 TB-ECDSA 方案在运算成本上低于主流的 ECDSA 门限签名方案。

3、基于 TB-ECDSA 方案实现了一个门限盲签名系统。该系统可以提供安全高效的私钥分发和多方计算盲门限签名服务。我们对系统进行了功能测试，以确保各项功能正常运行。接着，进行了性能测试，并将其与其他门限签名方案进行了对比。测试结果表明，TB-ECDSA 方案具有较高的算法效率。最后，我们进行了网络拥塞测试，评估了当某些参与者处于网络阻塞时对整个系统的耗时影响。

关键词：区块链，门限签名，门限盲签名，ECDSA，多方计算

ABSTRACT

In 2008, Satoshi Nakamoto introduced the concept of Bitcoin, which later emerged and gradually entered the public eye. With the maturation and application of Bitcoin technology, the management of digital assets faces increasingly severe security challenges. To address this issue, MPC (Multi-Party Computation) wallets emerged as a novel solution, enhancing the security of digital assets through private key splitting and decentralized management. However, the underlying technology of MPC wallets, ECDSA (Elliptic Curve Digital Signature Algorithm) threshold signature, poses challenges such as high computational costs and poor privacy. To tackle this problem, this paper conducts the following research:

1. A signature scheme named TB-ECDSA is proposed. Addressing the high computational costs and privacy issues of current ECDSA threshold signature schemes, we propose the TB-ECDSA scheme by combining threshold signatures and blind signatures. This scheme utilizes homomorphic encryption, ensuring signers cannot extract useful information during the signing process, thus preserving user privacy. Additionally, the TB-ECDSA scheme incurs lower computational costs compared to mainstream ECDSA threshold signature schemes.

2. The concept of user anonymity is introduced. To further enhance the privacy of multi-party computation systems, we aim to prevent intermediate data generated by multiple nodes serving multiple users in the system from being correlated. We introduce the concept of user anonymity, summarize the conditions and key points for achieving user anonymity, and use this basis to construct the TB-ECDSA scheme.

3. An algorithm for computing multi-party secret products is proposed, addressing the problem of computing the product of all secret values without revealing each party's secret value when each party holds a secret value. In this algorithm, nodes construct a new Shamir fragment by multiplying fragments of secret values, and by publicizing this fragment, each node can compute the product of all secret values.

4. A threshold blind signature system is constructed based on the TB-ECDSA

scheme. The system is built on an open-source cryptographic library and includes signer initialization modules, receiver initialization modules, and signature modules. We conduct functional tests on the system to ensure all functions operate normally. Subsequently, performance tests are conducted, comparing it with other threshold signature schemes. Test results indicate that the TB-ECDSA scheme exhibits higher algorithmic efficiency. Finally, network congestion tests are performed to assess the impact on the entire system's runtime when certain participants experience network congestion.

Key words: Blockchain, Threshold Signature, Threshold Blind Signature, ECDSA, Multiparty Computation

目　录

摘　要 ..... **I**
ABSTRACT ..... **II**
目　录 ..... **IV**
第一章 绪论 **1**
1.1 论文研究背景及意义 1
1.2 国内外研究现状 3
1.2.1 门限签名研究现状 3
1.2.2 盲签名研究现状 6
1.2.3 门限盲签名研究现状 8
1.3 本文主要研究工作 10
1.4 论文相关的结构安排 11
第二章 相关技术和理论介绍 **12**
2.1 Shamir 秘密共享 12
2.2 ECDSA 数字签名方案 12
2.3 Paillier 同态加密修改方案 13
2.3.1 Paillier 同态加密方案 13
2.3.2 Paillier 同态加密修改方案 14
2.4 判定性 Decisional Diffie-Helman(DDH)假设 15
第三章 基于多方计算的 ECDSA 门限盲签名方案 **16**
3.1 引言 16
3.2 用户匿名性 17
3.3 多方秘密乘积算法 18
3.4 角色定义 20
3.5 签名者初始化算法 21
3.6 接收者初始化算法 21
3.7 签名算法 22
3.7.1 阶段一 23
3.7.2 阶段二 25
3.7.3 签名验证 30
3.8 错误识别 30

3.9 安全性分析 32
3.9.1 不可伪造性 32
3.9.2 门限特性 32
3.9.3 盲性 33
3.9.4 用户匿名性 33
3.10 计算性能分析 35
3.11 本章小结 36
第四章 基于多方计算的 ECDSA 门限盲签名系统实现 38
4.1 开发环境搭建 38
4.1.1 Go 38
4.1.2 Protocol Buffer 38
4.1.3 tss-lib 39
4.2 系统功能实现 42
4.2.1 系统框架 42
4.2.2 签名者初始化模块 43
4.2.3 接收者初始化模块 45
4.2.4 签名算法模块 46
第五章 系统测试 51
5.1 功能测试 51
5.2 性能测试 53
5.3 网络拥塞测试 57
5.4 本章小结 61
第六章 总结与展望 63
6.1 总结 63
6.2 展望 64
参 考 文 献 65
附录 A 签名算法阶段二执行示意图 70
附录 B 签名算法阶段二执行示意图 71
致 谢 72
攻读硕士学位期间的科研成果 73

第一章 绪论

1.1 论文研究背景及意义

2008 年，中本聪提出了比特币[1]这一去中心化加密货币的设计概念。随着 2009 年比特币系统的启动，比特币正式问世。在 2010 年到 2015 年期间，比特币逐渐走入了公众视野。而 2016 年到 2018 年，各国相继对比特币公开表态，以及全球主要经济体不确定性的增加，这促使比特币备受关注，需求飞速增长。事实上，比特币可谓是区块链技术最为成功的应用之一。

随着以太坊[2]等开源区块链平台的涌现，以及大量去中心化应用的落地，区块链技术正被更多行业所应用[3]。随之而来的是越来越多的用户开始涉足区块链世界。在区块链世界中，每个用户对应一个公私钥对，公钥是用户在区块链世界中的身份，私钥则是用于生成数字签名。而数字签名是用户与区块链世界交互的主要方式，比如在比特币中用户使用私钥签署交易信息生成数字签名，再将其发布到区块链网络中，各个区块链节进行共识来完成一笔转账操作，其他更复杂的交互也需要基于数字签名，比如各种基于智能合约的去中心化应用也需要用户签署特定消息来引发相关的接口调用。因此掌握了私钥就等于掌握了数字资产的控制权。

为了使用户更便利地管理私钥和数字资产，数字钱包等产品迅速崛起，成为用户与区块链之间的纽带。数字钱包是一种能够存储、管理和交易加密数字资产的软件应用程序或硬件设备。它使得普通用户无需了解复杂的数字签名技术底层的密码学原理就能使用区块链平台提供的服务。~~这些钱包通常与区块链网络进行交互，使用户能够方便地发送、接收和管理诸如比特币、以太坊等加密货币。~~

数字钱包按照管理方式分类可以分为两类，自主控制钱包和托管钱包[4]。自主控制钱包由用户完全掌控自己的数字资产，需要自己保管好私钥或助记词，具有更高的安全性但是需要用户自行负责管理。托管钱包需要用户将数字资产或者私钥托管给第三方机构，方便快捷但存在风险。

随着区块链技术的普及和用户群体的扩大，这些方案的安全隐患也逐渐暴露出来：

1、对于使用自主控制钱包的用户，遗失私钥或私钥被黑客攻击是他们面临

的重大风险之一。许多用户或组织可能因个人疏忽，如存储私钥的不当或遗忘备份等，而导致私钥丢失，从而失去对其链上资产的所有权。根据《财富》杂志的数据统计，因各种原因导致持有者失去了对其地址的控制权造成丢失的比特币大约有 300 万个。

2、托管钱包的情况也不容乐观。将私钥托管给中心化管理的第三方机构，会造成数字资产高度集中，使得其攻击价值越来越高，从而成为黑客攻击的首要目标。2011 年，中心化数字货币交易所 Mt. Gox 遭受黑客攻击，丢失了近八十五万枚比特币，造成对大量用户造成了巨大的资产损失。

这些恶意事件的发生凸显了当前数字资产管理和交易过程中的安全性问题。用户对于自身数字资产的安全性越来越关注，迫切需要更加可靠和安全的解决方案来保护其资产免受损失。

多方计算(Multiparty Computation，MPC)钱包就是用户数字资产管理问题的一个解决方案。MPC 钱包作为一种新型的数字资产管理解决方案，相较于传统数字钱包具有明显的安全性优势。MPC 钱包通过私钥的拆分和分配，将私钥的控制权分散给不同的参与方，从而确保任何单一实体无法完全控制或访问该密钥。这种分布式的私钥管理模式大大降低了密钥被盗或泄露的风险，为用户提供了更加可靠和安全的数字资产管理环境。同时由于用户的私钥被分布式地保存在多个节点，MPC 钱包还可以有效规避单点故障的问题。通过使用多方计算技术，无论是为用户还是机构提供的钱包，都可以创建一个安全的链上资产管理系统，而无需使用单一的私钥。这消除了私钥被盗和丢失的风险，因为每个参与方都可以单独备份其秘数据，而无需暴露整个系统。

而门限签名方案是支持 MPC 钱包的关键技术。在 MPC 钱包中，私钥需要被分片并分配给参与方。不同的参与方持有私钥分片时，各参与方需要诚实地进行有效的协作才能安全地生成数字签名。这些需求都由门限签名方案解决。

在门限签名方案中，只有在达到一定数量的参与方合作下，才能生成有效的数字签名。这个门限值可以根据实际需求来确定，从而确保足够的安全性。通过这种方式，即使部分参与方受到了攻击或者行为不端，整个签名过程依然是安全可靠的。门限签名方案一般包含三个算法：密钥生成算法、签名算法以及签名验证算法[5]。这些算法通常是多个参与方的交互协议，通过一系列的多方计算和数据通信来共同计算出一个结果。

尽管门限签名方案能够在一定程度上抵御恶意参与者的影响，但当前的椭圆曲线签名(Elliptic Curve Digital Signature Algorithm, ECDSA)门限签名依然有以下问题：

1、运算成本高，如今主流的 ECDSA 门限签名方案涉及大量零知识证明和复杂的交互流程，使得算法复杂度与通信复杂度都比较高。

2、隐私性差，在现今的门限签名方案中，每次签名时，参与方不仅掌握私钥分片，还能够获取待签名消息的明文，这使得用户的交易行为完全暴露给了参与方，对用户的隐私性构成了威胁。

而另一种密码学技术——盲签名技术则很好的解决了隐私问题。在盲签名方案中，一般由两个参与方，一方是接收者，负责提供待签名消息，发起签名请求，最后接收数字签名。另一方是签名者，他持有私钥，负责完成数字签名的运算。消息在传达给签名者之前需要经过盲化处理，签名者只能对盲化的消息进行签名。当签名者完成签名后，接收者需要对受到的签名进行去盲，从而得到正确的签名。在盲签名方案中，签名者无法从签名的过程中得知关于签名的任何信息，也无法在公开网络中定位到哪个签名是出自自己的运算。然而在盲签名的方案中，大多数是私钥托管在一个可信方集中式的管理，因此存在着单点故障和过度中心化等问题。

本文的研究工作就是通过结合门限签名与盲签名两种方案，结合两者的有点，互补两者的缺点，提出一个既具有较低运算成本，又能保障用户隐私的签名方案，从而为构建更加安全高效的区块链资产管理系统提供支持。

1.2 国内外研究现状

1.2.1 门限签名研究现状

门限签名是一种密码学原语，旨在在多方参与的情况下生成和验证数字签名。传统的单一签名方案依赖于单个私钥持有者生成签名，这可能存在私钥被泄露或者单点故障的风险。为了解决这些问题，门限签名将私钥分割成多个部分，并要求多个签名者共同参与签名过程。

在门限签名方案中，私钥分割成多个部分，并分配给多个签名者。每个签名者持有私钥的部分，并且只有当足够数量的签名者参与签名并达到预设的门限值

时，才能生成有效的签名。这种多方参与的签名过程提高了安全性，因为即使部分私钥被泄露，攻击者也无法生成有效签名，因为他们没有达到门限值所需的私钥部分。

构建门限签名方案的最首要的问题是需要有一个可靠安全的方式进行私钥风分割。1979 年 Shamir 提出了一个秘密共享方案[6]612-613，一种基于多项式插值理论的技术，它为私钥分割提供了理论基础。在 Shamir 秘密共享方案中，将秘密值作为一个多项式的常数项，该多项式的函数值和对应的自变量就可以作为这个秘密的分片。当获得足够数量的秘密分片后，即可通过多项式插值的方式还原秘密值。然而使用 Shamir 秘密共享方案进行私钥分配需要一个可信方来作为私钥的分发者。为了解决这个问题，1991 年 Pedersen[7]提出了一种分布式密钥生成(Distributed Key Generation,DKG)算法，在算法种多个节点各自持有一个秘密值，节点首先通过广播他们的秘密值的 Shamir 秘密共享分片，然后每个节点将收到的分片相加，就得到了一个关于所有节点秘密值的求和的 Shamir 分片。DKG 算法的推出实现了无需信任方的分布式密钥生成，为各种门限签名方案提供了可行的基础。多年来，DKG 算法引起了广泛的关注和研究[8,9]，这些研究分析 DKG 算法中的安全风险，为之后 DKG 的改良和广泛应用打下了基础。

1987 年 Yvo Desmedt[10]指出当消息面向的是群体而不是个人时，传统的加密系统和公钥系统就变得不适用了，并从此出发分析了多个场景下的加密系统需求，其中包括了门限签名的概念。此后，在 2000 至 2004 年间出现了若干门限签名的方案[11,12,13,14]，这些方案这些方案在不同程度上解决了门限签名的问题。2003 年，王斌和李建华[15]提出了一个无可信中心的(t,n)门限签名方案，通过联合秘密共享技术，使得方案中无需一个所有成员都相信的诚信方也可完成签名。2014 年，尚铭[16]等人基于 SM2 椭圆曲线公钥密码算法，提出了一个安全有效的门限密码方案，该方案可以支持有可信方中心和无可信中心的两种情况。

随着区块链的应用不断扩展，对加密系统的需求变得更加迫切和复杂。在区块链系统中，数字签名是确保数据的完整性和身份验证的重要组成部分。因此，与数字签名相关的研究也受到了更多的关注和重视。特别是在区块链环境中，由于其去中心化和分布式特性，对数字签名的安全性和可靠性要求更高，一些新的、更安全和高效的数字签名技术得到了开发和探索。在这种情况下，基于 ECDSA 的门限签名成为研究的焦点之一。

Gennaro 和 Goldfeder 在 2018 年提出的 GG18[17]算法是首个高效的 ECDSA 门限签名方案。GG18 算法采用了一种基于可验证秘密共享（Verifiable Secret Share, VSS）[18]方案的分布式密钥生成协议，能够在无需信任方的情况下完成密钥分片的生成。此外，GG18 算法基于同态加密构建了多方协同计算的乘法转换加法协议，有效解决了多个参与方共同计算两组数字的求和并相乘的问题。GG18 算法的提出对于门限签名领域的发展具有重要意义，并为进一步的研究和应用奠定了基础。

Doerner 等人在 2019 年将同样由他们提出的 DKLS18[19]两方 ECDSA 算法拓展到(t,n)的一般情况，提出了 DKLS19[20]。DKLS18 在全局随机预言机模型下，该协议在 UC 框架中的安全性得到了证明，从而确保了与并发会话的安全性。虽然该协议中存在非常数次签名轮数，但可以将其转化为具有常数轮数的协议，尽管这可能会增加额外的通信开销。

在 2020 年，Castagnos 等人[21]提出了一种新的 GG18 协议的变体，该协议避免了所有所需的范围证明，并且在恶意对手的情况下保持了可证明的安全性。作者进行的实验表明，对于所有安全级别，他们的签名协议降低了先前已知安全协议的带宽消耗，降低因子在 4.4 到 9 之间，而密钥生成的成本始终少了两倍。此外，与这些相同的协议相比，他们的签名生成在 192 位及以上的安全级别上更快。同年，GG18 的作者以及另外三位研究者提出了 CGGMP 协议[22]，该方案是对 GG18 协议的再一次改进，在保持与 GG18 相当的计算成本的前提下，通过在每步计算中引入零知识证明以确保参与者忠实地执行协议增加了支持识别恶意参与者功能。

2023 年，Harry W. H. Wong 等人[23]提出了一种实际可行的门限 ECDSA 方案。该方案在预签名和签名阶段实现了真正的 t-out-of-n 门限值灵活性，无需假设存在诚实的多数方，适用于计算资源有限和存在分散故障的分布式环境。方案设计包括 4 轮预签名、O(n)作弊者识别和基于分布式份额的自我修复机制，可节省多达 30%的通信成本。与现有方案相比，该方案在自我修复和通信成本方面具有优势，是一种创新的阈值 ECDSA 方案。

ECDSA 门限签名协议旨在解决多个参与者共同完成符合 ECDSA 标准的数字签名的问题。然而，由于 ECDSA 的特性，ECDSA 门限签名协议需要复杂的协议和各种零知识证明来确保协议的安全性和可靠性。这使得大多数 ECDSA 门

限签名协议在计算和通信方面都具有较高的要求。在这方面，基于其他签名方案构建的门限签名就有很大的优势。

Schnorr 签名[24]具有线性，使用 Schnorr 签名的各方可以生成对其各自密钥的签名聚合，这使得 Schorr 签名非常适合用于构建多方签名方案。FROST[25]是一个门限签名协议，用于生成 Schnorr 签名。该协议的特点在于其密钥生成和签名过程仅需三轮通信，且通过预处理阶段的实施，签名过程可进一步优化至仅需一轮广播通信。与以往的 Schnorr 门限方案相比，FROST 在保持对已知伪造攻击安全性的同时，不会限制签名操作的并发性。 +?

1.2.2 盲签名研究现状

1982 年，Chaum[25]提出了第一个基于 RSA 的盲签名方案，并在不久后利用该方案构建了第一个电子货币方案[26]。Chaum 通过一个现实世界的例子来解释盲签名的概念：在内含有复写纸的信封上签名，使得签名同时被传输到信封内的纸片上，从而让收信人确认文件上有签名，但签名者无法知晓具体签署了哪些文件。在盲签名中，一般有两种角色，分别是接收者和签名者。接收者是指那些希望获取签名的个体或实体一般代表需要获取签名服务的用户，而签名者则是负责生成签名的一方。接收者首先对待签名的消息进行盲化处理，然后将处理后的消息发送给签名者进行签名。盲化处理的关键在于使签名方无法得知所签消息的具体内容，从而确保用户的隐私得到有效保护。

+++

1996 年，David Pointcheval 和 Jacques Stern[28]指出以往的盲签名方案都是基于传统签名方案，并额外添加了盲化证明。其中一些底层签名方案可以通过随机预言机模型的证明来验证其安全性，但原始签名方案的安全性本身并不能保证盲签名版本的安全性。因此他们提出了盲签名安全的定义，并对 Okamoto[29]的盲签名方案进行了安全性证明。

自盲签名技术提出以来，该领域受到了国内外学者的广泛关注和深入研究。学者们积极探索并提出了各种不同密码体制下的盲签名方案。这些方案在设计上考虑了不同的安全性需求和应用场景，以满足实际应用的多样性。

1995 年，Markus Stadler 等人[30]指出现有的盲签名方案都提供了完美的不可连接性，这样的特性可能存在滥用的风险，比如可能被犯罪分子用于勒索或洗钱等活动。为了解决这个问题，Markus Stadler 等人提出了一种公平盲签名(Fair

Blind Signatures)的方案。方案中存在一个可信实体可以提供信息，通过运行链接恢复协议，签名者可以从可信实体那里获取信息，使其能够识别相应的协议视图和消息-签名对。

另一方面 Masayuki Abe 和 Eiichiro Fujisaki[31]在 1996 年指出盲签名的不可链接性带来的另一个问题：签名者无法确定盲化的消息是否准确地包含了他所期望的信息。为了解决这个问题，他们提出了部分盲签名方案(Partial Blind Signatures)，其中消息的一部分保持明文，而其余部分保持秘密。该方案基于 RSA 加密算法，并证明了破解该方案与破解 RSA 一样困难。论文还讨论了将该方案应用于使用陷门函数的其他盲签名方案，并展示了一个最小化银行数据库大小的电子现金系统作为该方案的应用。

Georg Fuchsbauer 等人[32]在 2020 年对基于 Schnorr 签名的盲签名方案进行了严密的安全分析，他们证明了如果敌手要对 Schnorr 盲签名进行伪造攻击，那么他需要解决超定可解线性方程组中的随机不均匀性(Random inhomogeneities in a Overdetermined Solvable system of linear equations, ROS)问题[33]。他们对解决 ROS 问题的规约作出了准确定义，由于 ROS 问题可以使用 Wagner 算法在亚指数时间内解决，作者提出了对签名协议的简单修改，这些修改不会改变签名本身，因此与已经使用 Schnorr 签名的系统兼容。作者表明修改后的方案的安全性依赖于一个与 ROS 相关的问题，该问题更加困难。

自 2018 年以来，随着比特币等区块链技术的普及，基于 ECDSA 的盲签名方案也开始被提出。2019 年，Xun Yi 和 Kwok-Yan Lam[34]提出了一种可用于比特币交易匿名性的 ECDSA 盲签名方案。该方案对 Paillier 同态加密方案[35]229 进行了修改，并证明了修改后的 Paillier 同态加密方案的语义安全性。然后，他们基于修改后的 Paillier 同态加密方案构建了一个基于 ECDSA 的盲签名方案，并详细描述了如何将该方案应用于比特币的匿名交易。三年后，Xianrui Qin[36]等人进一步完善了对 ECDSA 盲签名方案的研究。他们构建了对任意 ECDSA 盲签名方案的攻击方案，提出了 ECDSA-ROS 问题假设，并完善了 ECDSA 盲签名方案的安全性分析。同时，他们还提出了一种 ECDSA 盲签名方案，并指出该方案的性能效率约为 Xun Yi 和 Kwok-Yan Lam 在 2019 年提出的方案的 40 倍。

2023 年，Lucjan Hanzlik[37]提出了一种新的盲签名改进方案称为非交互式盲签名。在签名时签名者创建一个预签名，包含接收者的可验证随机函数公钥和一

个随机数。接收者使用可验证随机函数对随机数进行评估，得到消息，并创建一个证明，证明其知道一个在给定的可验证随机函数密钥和随机数下的签名，并且消息是可验证随机函数评估的结果。在签名验证时，验证者使用可验证随机函数验证接收者提供的证明，并确保其符合要求。同年，陈倩倩与秦宝东提出了一种基于 SM9 的两方协同盲签名方案[38]，该方案通过利用密钥生成中心将 SM9 私钥分割成两个分片，并将其分配给两方签名者，从而使得签名者之间相互独立，确保了签名的安全性和可靠性。在签名的过程中，两方签名者通过一个协同盲签名协议完成签名，从而保障了签名的完整性和保密性。这一方案不仅考虑了用户隐私的保护，还同时确保了签名者的身份和行为信息不会泄露。需要指出的是，目前该方案仅适用于两方协同的情况，尚未涉及到多方协同的场景。

自 1982 年 Chaum 提出第一个基于 RSA 的盲签名方案以来，这一领域取得了显著进展。他的方案不仅奠定了盲签名的基础，还为电子货币的发展铺平了道路。随后，学者们不断完善盲签名方案的安全性和功能性，涌现出了多种新的方案。然而，随着技术的进步，也暴露出了盲签名可能被滥用的风险，例如被用于勒索或洗钱等不法活动。针对这些问题，学者们提出了公平盲签名和部分盲签名等新方案，以及对现有方案的安全性分析和改进。随着区块链技术的兴起，基于 ECDSA 的盲签名方案也逐渐被提出并改进，为比特币等加密货币的匿名交易提供了新的可能性。近年来，非交互式盲签名和基于 SM9 的两方协同盲签名方案等新技术的提出进一步拓展了盲签名的应用范围和安全性。这些创新不仅促进了盲签名技术的发展，也为数字身份认证和隐私保护等领域带来了新的思路和解决方案。

1.2.3 门限盲签名研究现状

门限盲签名技术融合了门限签名和盲签名的特性，实现了密码学上的创新。门限签名涉及将签名密钥分割成多个部分，各部分由不同的参与方持有，仅在达到门限值时才能合并生成有效签名，这确保了签名的分布式管理和控制。而盲签名则允许对消息进行盲化，保障了签名者和消息内容的匿名性和不可追踪性。在门限盲签名方案中，签名密钥被分割，并对消息进行盲化，只有在达到门限值时，多个参与方合作解盲化消息并生成签名，从而既实现了多方管理和控制，又保持了匿名性和不可追踪性。

2015 年，Veronika Kuchta 和 Mark Manulis[39]详细阐述了门限盲签名的概念，提出了门限盲签名方案的四个基本算法，即公共参数生成、密钥生成、门限盲签名以及门限盲签名验证。除此之外，他们还明确定义了门限盲签名应满足的两项主要性质:不可伪造性和盲性。

在 2023 年，Elizabeth Crites 等人提出了一种名为 Snowblind[40] 的门限盲签名方案，旨在克服现有盲签名方案的限制。Snowblind 方案构建在非门限盲签名的基础上，通过引入适当的框架成功实现了门限盲签名，并确保了其盲性质。在 Snowblind 方案中，用户不仅扮演接收者的角色，还承担了协调者的重要角色。在每一轮签名过程中，签名者之间并不直接通信，而是通过用户作为中转站传递协议消息。在签名协议结束时，用户聚合从签名者那里收到的签名分片，从而得到一个完整的签名，并将其发布出去。

正如 1.2.2 小节所述，盲签名方案普遍具备不可追踪性的特征，然而这种属性可能存在滥用的潜在风险，因此人们提出了公平盲签名。在门限盲签名中也有实现了公平盲签名特性的方案。Wen-Shenq Juang 和 Horng-Twu Liaw[41]在 2003 年提出了一个高效的门限公平盲签名方案，并在该方案中引入了观察者的概念。在一个特定的签名过程中，有三类参与者，分别是签名者、法官以及接收者。在接收者收到签名者发来的签名之前，所有的签名者都必须进行合作，提前将他们的一个秘密值分发给其他签名者。然后接收者将他的防篡改设备与法官配对，并使用钱包向签名者请求一个公平的盲门限签名。

在盲签名中，存在签名者无法确定盲化的消息是否准确地包含了他所期望的信息的问题，这一问题的解决方案之一便是引入部分盲签名方案。在门限盲签名领域，也有为了解决该问题而提出的门限部分盲签名特性的方案。Wen-Shenq Juang 和 CL Lei[42]基于离散对数困难问题提出了一个门限部分盲签名方案。该方案中，门限部分盲签名的大小与单个部分盲签名相同，而门限部分盲签名的验证过程则可通过群公钥进行简化。此外，该方案还具备消息恢复能力。方案的安全性建立在计算离散对数的困难性上，这使得对于签名者来说，在签名过程的执行中，计算出他们实际签名的消息与所有签名者完整视图之间的确切对应是计算上不可行的。同样提出门限部分盲签名方案的还有陆洪文等人[43]，他们提出了一种基于双线性对的新型的门限部分盲签名方案。该方案通过加入签名方的信息控制发送方，并引入双线性对的概念，限制了由单个签名者组成的系统的不安全性。

门限盲签名技术将门限签名和盲签名的优势相结合，实现了密码学上的重要创新。该技术通过将签名密钥分割并对消息进行盲化，使得多方参与者在达到门限值时才能生成有效签名，从而实现了签名的分布式管理和控制，并保护了签名者和消息内容的匿名性和隐私。然而在门限盲签名领域却很少有基于 ECDSA 的门限盲签名方案的研究，这使得门限盲签名的应用受到了一定局限。

1.3 本文主要研究工作

ECDSA 是一种广泛采用的数字签名方案，在实际项目中已经有许多 ECDSA 门限签名方案被研究并在 MPC 钱包项目中得到应用[44]。另一方面，基于 ECDSA 的盲签名也在一定程度上得到了发展。然而，但基于 ECDSA 的门限盲签名方案的研究相对较少。在 ECDSA 广泛应用的区块链领域，用户私钥的管理问题以及用户安全性和隐私性的问题仍然在等待更好的方案来给予解决。为了给解决以上问题提供支持，本文进行了以下工作：

1、我们提出了一个基于多方计算的 ECDSA 门限盲签名方案，该方案具有门限特性、盲性、不可链接性、错误识别以及用户匿名性的特性。为了进一步提升隐私性，我们引入了用户匿名性的概念，并通过场景分析总结了实现用户匿名性的条件。接着，我们提出了一个多方秘密乘积算法，用以解决多节点间安全计算秘密值相乘的问题。基于用户匿名性的实现条件，我们借助所提出的多方秘密乘积算法，结合 Paillier 同态加密系统，设计了一个基于多方计算的 ECDSA 门限盲签名方案。最后，我们对该方案的安全性和计算性能进行了理论分析。

2、针对用户私钥管理、安全性和隐私性问题，我们基于提出的方案设计并实现了一个基于多方计算的 ECDSA 门限盲签名系统，旨在为用户提供安全、隐私的密钥生成和数字签名服务。该系统的构建利用了开源密码库 tss-lib 提供的密码学工具，采用了 Protocol Buffer 定义消息类型，并采用 Go 语言进行了开发。

3、对于本文提出的基于多方计算的 ECDSA 门限盲签名系统，我们从三个方面进行了测试，分别是功能测试、性能测试以及网络拥塞测试。对于功能测试，我们设计了测试模板程序，在该程序下运行不同模块的程序并验证了其输出结果的正确性。在性能测试中我们将该系统与其他门限签名方案进行了对比，测试了在不同算法下各方案的运行耗时，测试结果表明，本文提出的方案具有较高的算法效率。最后，我们进行了网络拥塞测试，评估了当某些参与者处于网络阻塞时

对整个系统的耗时影响。

1.4 论文相关的结构安排

本文总共由五个章节组成，每个章节的内容安排如下：

第一章，绪论。本章介绍了本研究的背景与研究意义，讲述了当前区块链应用中，用户以及组织面临的私钥管理问题并阐述了门限签名、盲签名和门限盲签名三项技术的国内外研究现状。本章的最后两小节介绍了本文的研究工作以及论文的结构安排。

第二章，相关技术与理论介绍。本章主要介绍了构建本方案所需的密码学工具，包括 Shamir 秘密共享、ECDSA 数字签名方案、Paillier 同态加密修改方案、判定性以及 Decisional Diffie-Helman 假设。这些理论和技术为后续章节的方案设计提供了基础。

第三章，基于多方计算的 ECDSA 门限盲签名方案。本章首先介绍了该方案具有的几个特性，然后提出了用户匿名性的概念和一个计算多方秘密乘积的算法。接着详细介绍了方案中设计的角色，并逐步解释了方案中包含的三个算法的步骤和计算原理。最后对方案的安全性进行分析和证明，并从理论上分析了方案的性能。

第四章，基于多方计算的 ECDSA 门限盲签名系统。本章针对第三章提出的方案进行了系统实现，首先介绍了开发环节的搭建，接着介绍了从系统框架到各个模块的实现。

本章对第四章的系统进行了三个方面的测试，分别是功能测试、性能测试和网络拥塞测试。通过这些测试综合评估了系统的特点。

第六章，总结与展望。本章对本文的工作进行总结，回顾了研究的主要成果和贡献，并对未来改进方向进行了讨论和展望，为进一步研究提供了参考和启示。

第二章 相关技术和理论介绍

2.1 Shamir 秘密共享

Shamir 秘密分享[6]612-613 是由 Adi Shamir 提出的一个算法。该算法可以将一个秘密值分成 n 个分片，当得知的分片数量大于一个阈值时，才可以从分片中还原出秘密值。基本原理是利用多项式插值。具体来说，原始秘密被表示为一个多项式的常数项，多项式的次数取决于设定的分片数量阈值。然后，多项式在不同的点上被计算得到秘密分片值，这些分片值被分发给不同的参与者。当需要恢复原始秘密时，只需收集足够数量的秘密分片值，然后使用插值技术即可重建出原始秘密。

对于一个秘密值ε，以ε为常数项构造一个t次多项式$f(x)$:

$$f(x)=\varepsilon+a_1x+a_2x^2+...+a_tx^t \tag{2-1}$$

则对于不同的x取值，如$x_1,x_2,...,x_n$可以计算出不同的函数值$f(x_1),f(x_2),...,f(x_n)$，这些函数值与其自变量共同构成一个秘密分片值$(x_i,f(x_i))$，当收集到 t+1 个分片值时，可以对 t+1 个分片通过以下运算重建原始秘密:

$$\Gamma_i=\prod_j\frac{x_i}{x_j-x_i} \tag{2-2}$$

$$\varepsilon=\sum_i\Gamma_if(x_i) \tag{2-3}$$

2.2 ECDSA 数字签名方案

数字签名技术是公钥加密技术的一种应用[45]，在数字签名方案中有公钥和私钥，公钥公开，而私钥由签名人保密持有。签名人可以对某个消息利用私钥进行加密运算，这个过程为签名，加密后得到的数据称作数字签名。任何人都可以使用公钥来验证该数字签名是由该签名人签署的。

椭圆曲线签名算法(Elliptic Curve Digital Signature Algorithm, ECDSA)[46]是数字签名算法(Digital Signature Algorithm , DSA)的一种变体，在区块链和加密货币中受到广泛应用。ECDSA 数字签名方案包括四个部分。

1、公共参数：

（1）一个阶为q的椭圆曲线循环群G，以及其生成元g，其中q为素数。

（2）哈希函数H，可以将任意类型的消息 M 映射到Z_q

2、密钥生成算法：输入一个安全参数，输出一对公私钥。私钥为从Z_q中随机选取的整数x，公钥为$y=g^x$。

3、签名生成算法：输入一个任意的消息M,输出签名$\sigma=(r,s)$。

（1）计算$m=H(M)\in Z_q$。

（2）随机选择整数$k\in_R Z_q$。

（3）计算$K=g^k\in G$，取K的横坐标为r。

（4）计算$s=k^{-1}(m+xr)\bmod q$。

（5）输出$\sigma=(r,s)$。

4、签名验证算法：输入M，σ和y。

（1）验证r和s是否都属于Z_q。

（2）计算$R'=g^{ms^{-1}\bmod q}y^{rs^{-1}\bmod q}$。

（3）取R'的横坐标为r'，如果$r'=r$则签名验证通过，否则该签名无效。

2.3 Paillier 同态加密修改方案

2.3.1 Paillier 同态加密方案

Paillier 同态加密方案[35]229是一个基于判定性合数剩余假设[47]构建的加密方案，它具有加法同态的性质。

1、密钥生成算法：

（1）随机选取两个大素数p,q，需要满足条件pq与$(p-1)(q-1)$互素，不满足则重新选择随机数。

（2）计算$N=pq$，$\lambda=lcm(p-1,q-1)$，其中lcm表示最小公倍数。

（3）随机选取一个正整数g，要求$g<N^2$并存在$\mu=(L(g^{\lambda}\bmod n^2))^{-1}\bmod n$，若$\mu$不存在，则需要重新选取$g$,其中：

$$L(x)=\frac{x-1}{N} \tag{2-4}$$

公钥为(N,g)，私钥为(p,q)。

2、加密算法：

对于要加密的消息m，它需要满足$0 \le m \le N$，它的加密过程如下：

随机选取一个正整数r，要求$0 < r < N$，且还需满足r与N互素。计算密文c如下：

$$c = g^m r^N \bmod N^2 \tag{2-5}$$

3、解密算法：

对于密文c，解密过程为：

$$m = L(c^\lambda \bmod N^2)\mu \bmod N \tag{2-6}$$

4、加法同态性质：

Paillier 加密具有加法同态的性质，使用$E(m)$表示对m进行加密得到的密文，对于任意的消息a,b，加法同态表示为：

$$E(a)E(b) \bmod N^2 = E(a+b) \tag{2-7}$$

$$E(a)^b \bmod N^2 = E(ab) \tag{2-8}$$

将$\oplus$定义为 Paillier 的同态加运算，$\otimes$定义为 Paillier 的同态数乘运算，则上述性质可以表示为：

$$E(a) \oplus E(b) = E(a+b) \tag{2-9}$$

$$b \otimes E(a) = E(ab) \tag{2-10}$$

2.3.2 Paillier 同态加密修改方案

Paillier 同态加密因其同态加密的性质而得到广泛应用，然而在文献[34]616 中提到将 Pailiier 方案直接应用于构建盲 ECDSA 签名可能导致更多信息的泄露，而这部分信息的泄露所带来的安全性影响是未知的。因此文献[34]616-617 在此基础上提出了一个 Paillier 同态加密的修改方案，并证明了它在选择明文攻击下是语义安全的。

1、密钥生成算法：

（1）随机选取两个大素数p,t，需要满足条件pt与$(p-1)(t-1)$互素，不满足则重新选择随机数。

（2）计算$N = pqt$，$g = (1+N)^{pt} \bmod N^2$，其中q为 ECDSA 中的公共参数椭圆曲线群的阶。得到的公钥为(N,g)，私钥为(p,t)。

2、加密算法：

（1）选择要加密的消息 m，它需要满足 $0 \le m \le q$

（2）随机选取一个正整数 r，要求 $0 < r < N$，且还需满足 r 与 N 互素。

计算密文 c 如下：

$$c = g^m r^N \bmod N^2 \tag{2-11}$$

3、解密算法：

对于密文 c，解密过程为：

$$m = \frac{c^{(p-1)(q-1)(t-1)} \bmod N^2 - 1}{Npt} [(p-1)(q-1)(t-1)]^{-1} (\bmod q) \tag{2-12}$$

对于修改后的 Paillier 同态加密方案，同样具有同态加法性质，即密文的相加等于明文相加后再加密。为了方便表达，本文第三章中提到的 Paillier 同态加密方案指的都是 Paillier 同态加密的修改方案。

2.4 判定性 Decisional Diffie-Helman(DDH)假设

判定性 DDH 假设[48]是一个密码学中的一个重要数学假设，在数字签名、承诺和数字加密方案中有着广泛应用。判定性 DDH 假设的定义如下：

给定阶为 q 的循环群 G，其中 q 是一个大素数，g 为循环群 G 的生成元，无法在多项式时间内区分以下两个四元组 R 和 D，其中 $a,b,c \in Z_q$：

$$R = \left(g, g^a, g^b, g^c\right) \tag{2-13}$$

$$D = \left(g, g^a, g^b, g^{ab}\right) \tag{2-14}$$

第三章 基于多方计算的 ECDSA 门限盲签名方案

3.1 引言

在本章中，我们提出了门限盲签名方案 TB-ECDSA（Threshold Blind ECDSA）。该方案涉及两个主要参与角色：接收者和签名者。我们假设接收者是用户本身，具有诚信性，而签名者可能存在恶意行为。在这种前提下，我们设计了 TB-ECDSA 的三个算法，包括签名者初始化算法、接收者初始化算法和签名算法。本文提出的 TB-ECDSA 方案具有以下几个特点：

门限特性：TB-ECDSA 方案中允许有 n 位签名者，当 t 位签名者同意参与签名时才可以产生合法的签名，其中 $n>3, t$ 满足 $n/2<t<n$。签名者必须达成共识并同意参与签名操作才能产生合法的签名。这种门限签名机制确保了多个签名者的合作，增加了签名的可靠性和安全性。

盲性和不可链接性：在签名过程中，签名者无法获取与签名相关的信息，也无法确定签名过程与后续公开签名之间的确切关系。借助同态加密技术，签名者只能对同态加密后的密文进行同态运算，而无法揭示原始消息的内容。这种特性不仅保护了签名的机密性和隐私性，还有效地防止签名者通过中间数据获取额外信息的可能性，从而确保了签名者无法追溯到具体的签名行为。因此，接收者的隐私得到了保障，免受潜在的泄露风险的威胁。

错误识别：在签名算法中，如果签名者违反了算法规定的计算方式，导致最终产生的签名不合法，接收者可以通过分析签名过程中产生的中间变量和公开的参数来追溯是哪个签名者的错误行为导致了签名的无效。通过对比签名者之间的计算结果和公开参数，接收者可以定位并识别出签名过程中的错误行为，从而可以追究责任或采取相应的措施来防止类似错误的再次发生。这种错误识别机制增强了签名算法的安全性和可靠性，确保签名者按照规定的方式进行签名，从而保护签名的有效性和准确性。

用户匿名性：在签名过程中，即使存在多个签名者共谋，它们也无法确定特定数据是否来自同一个用户。这种匿名性保证了用户的身份和行为无法被追踪或识别。此外，由于用户匿名性的存在，签名者之间也无法确定私钥与特定用户之间的对应关系，这增加了共谋者窃取私钥的难度，进一步保护了用户的隐私和匿

名性。因此，用户匿名性在签名过程中为用户提供了更高级别的隐私保护，防止了身份泄露和数据关联。

3.2 用户匿名性

在本节，我们提出用户匿名性特性，并总结了具有用户匿名性的多方计算系统需要受到的规则约束。

用户匿名性是指在一个用户请求一个多方计算服务的过程中，当存在多个节点共谋时，节点无法确定特定数据是否来自同一个用户。用户匿名性可以由以下例子进行说明。假设有用户 R_1 和 R_2，节点 S_1 和 S_2，用户 R_1 和 R_2 都向两位签名者请求了相同的服务，两个节点依照一定协议完成了服务。设整个服务过程中 R_i 向 S_j 请求服务产生的所有中间数据为 D_{ij}。此时 S_1 持有 (D_{11}, D_{21})，S_2 持有 (D_{12}, D_{22})。当某个攻击者 A 控制了 S_1 和 S_2 获得了所有的数据，此时 A 持有 $(D_{11}, D_{21}, D_{12}, D_{22})$，如果 A 无法根据数据的内容确定四组信息中的哪两组是来自同一个用户，则该系统满足用户匿名性特性。

为了实现用户匿名性，在设计多方计算服务流程时需要受到以下规则约束：

1、在一次服务中多个节点不能发送含相同内容的数据。

2、在一次服务中多个节点不能接收含相同内容的数据。

3、在一次服务中多个节点不能计算出含相同内容的数据。

4、在一次服务中节点不能直接向另外一名节点发送数据。

当上述规则中的任意一条无法满足时，签名者共谋时就可以将相关的数据匹配到对应的用户。我们对这四条规则进行详细的说明。

假如规则 1/2/3 其中之一不满足，在一次服务中多个节点发送/接收/计算出了相同的内容。假设节点 S_1 和 S_2 在为 R_1 提供服务的过程中发送/接收/计算出了相同的内容 η_1，为 R_2 服务的过程中发送/接收/计算出了相同的内容 η_2。则攻击者 A 就可以从数据集合 $(D_{11}, D_{21}, D_{12}, D_{22})$ 中的各项数据中提取到对应的 $(\eta_1, \eta_2, \eta_1, \eta_2)$。那么 A 就可以根据 $\eta_1 \in D_{11}$ 和 $\eta_1 \in D_{12}$ 确定 D_{11} 与 D_{12} 是来自同一个用户的请求产生的数据，根据 $\eta_2 \in D_{21}$ 和 $\eta_2 \in D_{22}$ 确定 D_{21} 与 D_{22} 是来自同一个用户的请求产生的数据，从而导致系统无法满足用户匿名性。

假如规则 4 不满足，在一次服务中，节点向另一节点直接发送了数据。假设节点 S_1 和 S_2 在为 R_1 提供服务的过程中，S_1 向 S_2 发送了数据 η_1，在为 R_2 提供服务

的过程中，S_1 向 S_2 发送了数据 η_2。则攻击者 A 就可以从数据集合 $(D_{11},D_{21},D_{12},D_{22})$ 中的各项数据中提取到对应的 $(\eta_1,\eta_2,\eta_1,\eta_2)$。那么 A 就可以根据 $\eta_1 \in D_{11}$ 和 $\eta_1 \in D_{12}$ 确定 D_{11} 与 D_{12} 是来自同一个用户的请求产生的数据，根据 $\eta_2 \in D_{21}$ 和 $\eta_2 \in D_{22}$ 确定 D_{21} 与 D_{22} 是来自同一个用户的请求产生的数据，从而导致系统无法满足用户匿名性。

上述规则是一个系统满足用户匿名性的必要条件，在满足该条件的系统中，节点在每次服务时只允许与用户交互，当一个节点需要和其他节点交换数据时，只能通过用户中转。用户匿名性的关键在于节点无法将同一次服务过程的数据相互关联起来。因此在系统中，用户与节点交互的数据必须具有计算不可区分性[49]，当用户中转其他节点的数据时，也需要对数据处理，使其具有计算不可区分性后才可以发送给下一个节点。

3.3 多方秘密乘积算法

为了构建基于多方计算的 ECDSA 门限盲签名方案，本文提出一种基于 Shamir 秘密分享的多方秘密乘积(Multi-party Secret Product Algorithm, MSP)算法。在 DKG 算法中，多个节点各自持有一个秘密值，节点首先通过广播他们的秘密值的 Shamir 秘密共享分片，然后每个节点将收到的分片相加，就得到了一个关于所有节点秘密值的求和的 Shamir 分片。

受 DKG 算法启发，我们提出了 MSP 算法。在 MSP 算法中，多个节点各自持有一个秘密值，借助一个无秘密值的辅助节点，通过 MSP 算法，所有节点都可以在不泄露秘密值的情况下计算出所有秘密值的乘积。设辅助节点为 S_c，其余节点为 S_i，$i=\{1,2,..,t\}$，每个 S_c 或 S_i 都有一个正整数 id_i 或 id_c 作为自己的唯一标识，每个算法各自持有的秘密值为 p_i，算法流程如下：

1、S_i 以 p_i 为常数构造一个一次多项式 $f_i(x)=p_i+a_ix$，其中 a_i 是随机选取的系数。计算 $p_{ii}=f_i(id_i)$，暂存 p_{ii}。计算 $p_{ij}=f_i(id_j)$ 和 $p_{ic}=f_i(id_c)$，将 p_{ij} 发送给 S_j，将 p_{ic} 发送给 S_c。

2、S_i 或 S_c 收到其他节点发送来的 p_{ji} 或 p_{jc}，计算：

$$p'_i=\prod_j p_{ji} \tag{3-1}$$

或

$$p'_c = \prod_j p_{jc} \tag{3-2}$$

3、S_i 或 S_c 计算 $p_{mi} = \Gamma_i p'_i$ 或 $p_{mc} = \Gamma_c p'_c$。其中:

$$\Gamma_i = \prod_j \frac{id_i}{id_i - id_j} \tag{3-3}$$

4、S_i 或 S_c 公布 p_{mi} 或 p_{mc}。

5、S_i 或 S_c 收到 p_{mi} 和 p_{mc} 后计算 $p = \sum p_{mj} + p_{mc}$，其中：

$$p = \prod p_i \tag{3-4}$$

我们对 MSP 算法进行一些补充说明，在步骤 2 中计算的 p'_i（p'_c 同理)可以展开成如下形式：

$$p'_i = \prod_j (p_j + a_j id_i) = \prod_j p_j + A_1 id_i + A_2 id_i^2 + \ldots + A_t id_i^t \tag{3-5}$$

其中 A_t 是由 p_j 和 a_j 这两个常数组合相乘构成的常数，我们并不关心 A_t 的具体值。式(3-5)的形式可以看作是一个多项式函数 $F(x)$ 在自变量为 id_i 时的取值，并且这个多项式的常数项为 $\prod p_i$，即如下式：

$$F(x) = \prod_j p_j + A_1 x + A_2 x^2 + \ldots + A_t x^t \tag{3-6}$$

$$p'_i = F(id_i) = \prod_j p_j + A_1 id_i + A_2 id_i^2 + \ldots + A_t id_i^t \tag{3-7}$$

因此可以将 p'_i 或 p'_c 看作是关于 $\prod p_i$ 的一个 Shamir 秘密共享分片，它的阈值为 $t+1$。然后在步骤 3 中，利用 Γ_i 将 p'_i 或 p'_c 转换乘了加法分片 p_{mi} 或 p_{mc}。在所有节点在步骤 4 公布自己的加法分片，而 p_{mi} 或 p_{mc} 中是不会泄露各个节点的 p_i 的最后各个节点就可以通过对 p_{mi} 和 p_{mc} 求和来得到秘密乘积 p。图 3-1 为一个辅助节点与三个节点进行一次 MSP 算法的执行图。

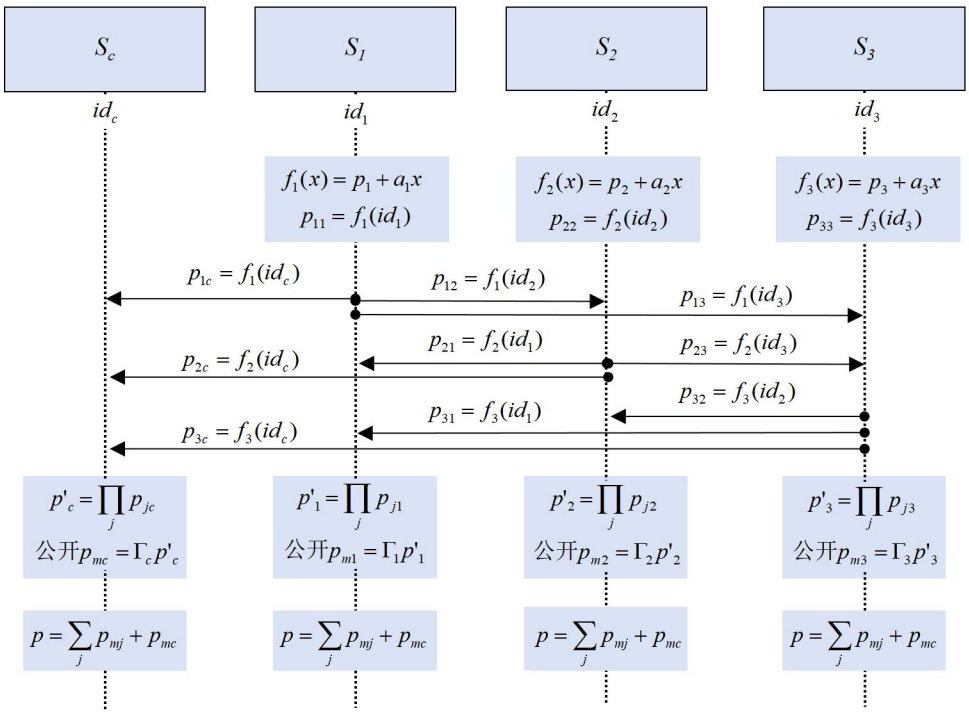


图 3-1 MSP 算法执行示意图

3.4 角色定义

在整个 TB-ECDSA 方案中，有两种角色：接收者和签名者。

接收者：接收者用符号 R 表示，他代表了使用 TB-ECDSA 的用户。接收者是最初发起签名请求的人，并最终接收到签名分片并完成聚合。在 TB-ECDSA 方案中，我们将接收者视为可信的实体。在进行签名算法之前，接收者会使用私钥作为秘密构造 Shamir 秘密共享，并将私钥分片分发给签名者。在签名算法过程中，接收者提供待签名的消息，并参与签名的生成过程，作为签名者之间的消息中转，以避免签名者之间直接联系破坏用户的匿名性。完成签名后，签名者会检查签名的合法性。如果签名不合法，接收者会对签名的中间数据进行排查，以识别违反协议的签名者。

签名者：签名者用符号 S_i 表示，下标 i 用于区分不同签名者。在 TB-ECDSA 中，n 个签名者组成签名者集合。签名者是签名算法的执行者，并拥有私钥分片。在 TB-ECDSA 方案中，签名者被视为不可信的实体。在进行签名算法之前，签名者需要以 t 个签名者为一组进行初始化，生成代表该组签名者的相关私有或公共参数。在进行签名算法时，签名者使用私钥分片和随机数与接收者交互逐步计算签名分片。

3.5 签名者初始化算法

每个签名者 S_i 都要在初始化的过程中得到一个属于自己的掩盖因子分片 p_i，以及一个公共参数掩盖因子 p，其中 p 为所有 p_i 的乘积。一次签名者初始化算法涉及 $t+1$ 个签名者，t 为方案设置的门限，这 $t+1$ 位签名者通过 MSP 算法完成掩盖因子分片你和掩盖因子的生成，一次初始化算法步骤如下：

1、确定一位签名者作为 MSP 算法的辅助节点 S_c，其余节点为 S_i，$i=\{1,2,..,t\}$。

2、S_i 选择一个大素数 p_i 作为掩盖因子分片。

3、S_i 以各自的 p_i 与 S_c 共同进行 MSP 算法。

4、S_i 得到公共参数 p。计算 $P_i=g^{p_i}$，公开 (P_i,p)。

一次初始化流程所产生的 p_i 和 p 与当前参与流程的签名者集合 S 对应，因此对于 n 个签名者，阈值为 t 的系统，需要进行 C_n^t 次初始化过程。各个签名者持有 C_n^t 个 p_i 代表它与不同签名者组成签名者集合时应该使用的掩盖因子分片。以一个协调签名这与三个签名者进行一次签名者初始化算法为例，图 3-2 是一次签名者初始化算法的执行示意图。

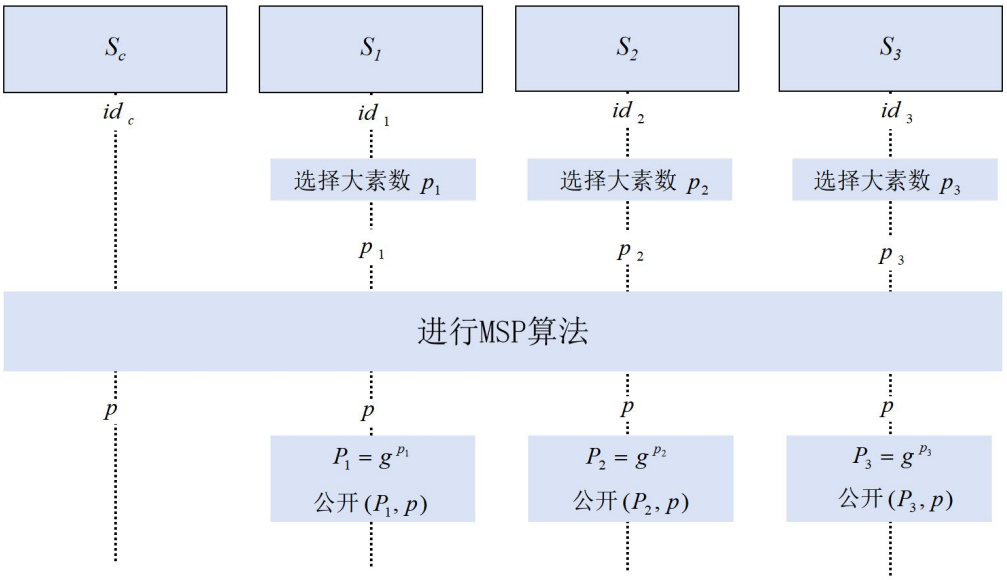


图 3-2 签名者初始化算法

3.6 接收者初始化算法

接收者需要准备 t 个 Paillier 加密方案的公私钥对，其对应的加密和解密操作记作 E_i 和 D_i。

接受者在请求签名者签名前需要将私钥以秘密共享的方式将私钥分片发送给签名者，签名者将私钥分片保存好，向接收者返回存储索引，接收者需要保存存储索引，以便在签名过程中使用。

接受者 R 拥有私钥 x 和对应公钥 y，接收者密钥分发步骤如下：

1、R 以 x 为秘密构造一个 Shamir 秘密共享，阈值为 t，秘密分片为 u_i。

2、R，分别将 u_i 发给 S_i。

3、S_i 接收到 u_i 后，存储 u_i 并发送向 R 发送存储索引 $index_i$。

4、R 保存索引 $index_i$，计算并保存 $U_i = g^{u_i}$，删除 u_i，将 x 秘密备份完成接收者初始化算法。

以一个接收者与三位签名者完成接收者初始化算法为例，图 3-2 是接收者初始化算法的执行示意图。

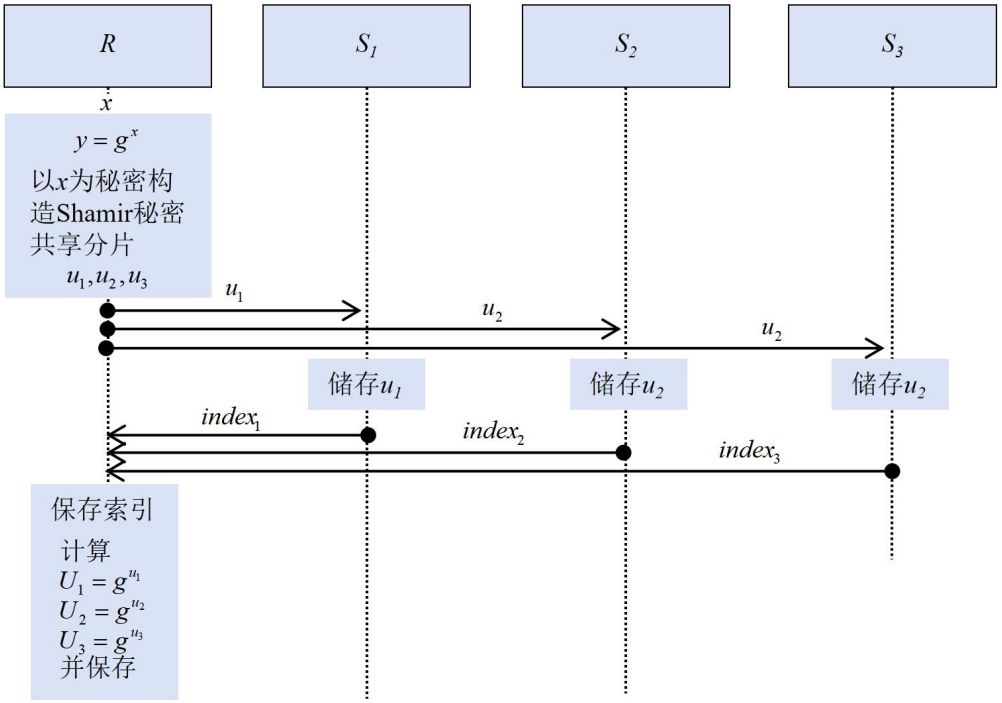


图 3-3　接收者初始化算法执行示意图

3.7 签名算法

签名算法是 TB-ECDSA 方案中最核心的部分，在签名算法中，由接收者提供待签名的消息 m，然后接收者与签名者共同计算出符合 ECDSA 数字签名方案的数字签名 σ。为了满足用户匿名性的四个约束规则，签名算法中接收者与签名者的交互有以下特点:

1、签名者只与接收者来往数据。

2、接收者收到的所有数据都会经过处理后再发送给下一个签名者。

3.7.1 阶段一

签名算法的阶段一目的是为了生成椭圆曲线签名中的随机点 K，在这个过程中只有接收者最终得知了 K，签名者参与 K 的生成但是无法得知 K 的真实值。

表 x-x 签名算法阶段一相关参数

公共参数	签名者私有参数
椭圆曲线群的相关参数 (G,g,q)	掩盖因子分片 p_i

表 x-x 为签名算法阶段一相关参数信息，算法步骤如下：

1、R 选择一个随机数 $k_r \in Z_q$，随机数 $\alpha \in Z_q$。

2、R 计算 $K_r = g^{k_r}$。

3、R 计算 $K_r' = K_r^{\alpha}$。

4、R 发送 (K_r, K_r') 给 S_1。

5、S_1 接收到 K_r，K_r' 后，随机选择一个 $k_1 \in Z_q$，计算 (K_{r1}, V_1, K_{C1}) 并将 (K_{r1}, V_1, K_{C1}) 发送给 R：

$$K_{r1} = K_r^{k_1} \tag{3-7}$$

$$V_1 = K_r'^{k_1} \tag{3-8}$$

$$K_{C1} = g^{k_1 p_1^{-1} \bmod q} \tag{3-9}$$

6、R 接收到 K_{r1}, V_1 和 K_{C1} 后，计算 $V'_1 = K_{r1}^{\alpha}$，如果 $V'_1 = V_1$ 成立，继续协议，否则中止协议。

7、R 计算 $K_{r^2 1} = K_{r1}^{k_r}$，将 $K_{r^2 1}$ 作为步骤 2 中的 K_r，对其他签名者依次进行步骤 3~6 直到与所有的签名者都交互一次。

8、完成步骤 7, R 会得到

$$K_{r^t 12\ldots t} = g^{k_r^t \prod_i k_i} \tag{3-10}$$

$$V_t = K_{r^t t}^{\alpha} \tag{3-11}$$

$$K_{Ct} = g^{p_t^{-1} k_t \bmod q} \tag{3-12}$$

9、 R 计算 $K = K_{r^t 12...t}{}^{k_r}$。

我们对阶段一的步骤作详细的解释。步骤 1~4 中，接收者 R 选择了一个随机数 k_r，然后接收者 R 对随机数做倍点运算得到签名随机点的中间结果 K_r，同时接收者 R 再用另外一个随机数 α 与 K_r 倍点运算得到 K_r'，然后将 K_r 和 K_r' 发送给一个签名者 S_1。

步骤 5 中，签名者 S_1 接收到相关数据后将他选的随机数 k_1 对 K_r 及 K_r' 做相同的倍点运算得到 K_{r1} 和 V_1，再将随机数 k_1 与掩盖因子分片的逆 p_1^{-1} 的乘积做倍点运算得到 K_{C1}，再将 K_{r1}, V_1 和 K_{C1} 发给接收者 R。

步骤 6 中，接收者 R 收到数据后，计算 $V'_1 = K_{r1}{}^{\alpha}$，验证 $V'_1 = V_1$。因为 K_r 与 K_r' 之间有数学关系 $K_r' = K_r{}^{\alpha}$，假如签名者对 K_r 及 K_r' 没有作相同的运算，或者偷换了其中的数据就会导致 $V'_1 = V_1$ 不成立，证明签名者 S_1 偏离了协议，没有依照协议对 K_r 做指定的运算，此时中止当前协议。如果验证通过了则继续进行协议，接收者保存 K_{C1}，以便进行后续的验证。

步骤 7 中，签名者 R 在将中间结果 K_{r1} 发送给下一个签名者继续处理前，需要用自己的随机数 k_r 与这个结果做倍点运算，然后再发送给下一位签名者。这样做是为了混淆前后两个签名者发送或者接受的数据，使后收到数据的签名者无法推测出上一个处理该数据的签名者是谁，避免他们将运算中间结果联系起来推测出更多关于接收者的信息。接收者 R 对 K_{r1} 完成处理后，发送给下一个签名者。如此往复直到所有的签名者都将自己的随机数与 K_{r1} 进行了倍点运算。此时接收者 R 得到的依旧是一个中间结果，需要对这个结果使用 k_r 再做一次倍点运算才得到生成椭圆曲线签名需要的随机点 K。经过上述协议后得到的 K 的具体值为：

$$K = g^{k_r{}^{t+1} \prod_i k_i} \tag{3-13}$$

那么 K 对应的 k 为：

$$k = k_r{}^{t+1} \prod_i k_i \tag{3-14}$$

虽然在步骤 4 中首先将数据发送给了 S_1 处理，实际上接收者 R 与签名者的交互顺序是可以任意的，只要保证所有签名者都将自己的随机数与中间结果作了倍点运算即可。以接收者与三个签名者共同进行阶段一为例，图 3-3 是进行一次阶段一的执行示意图。

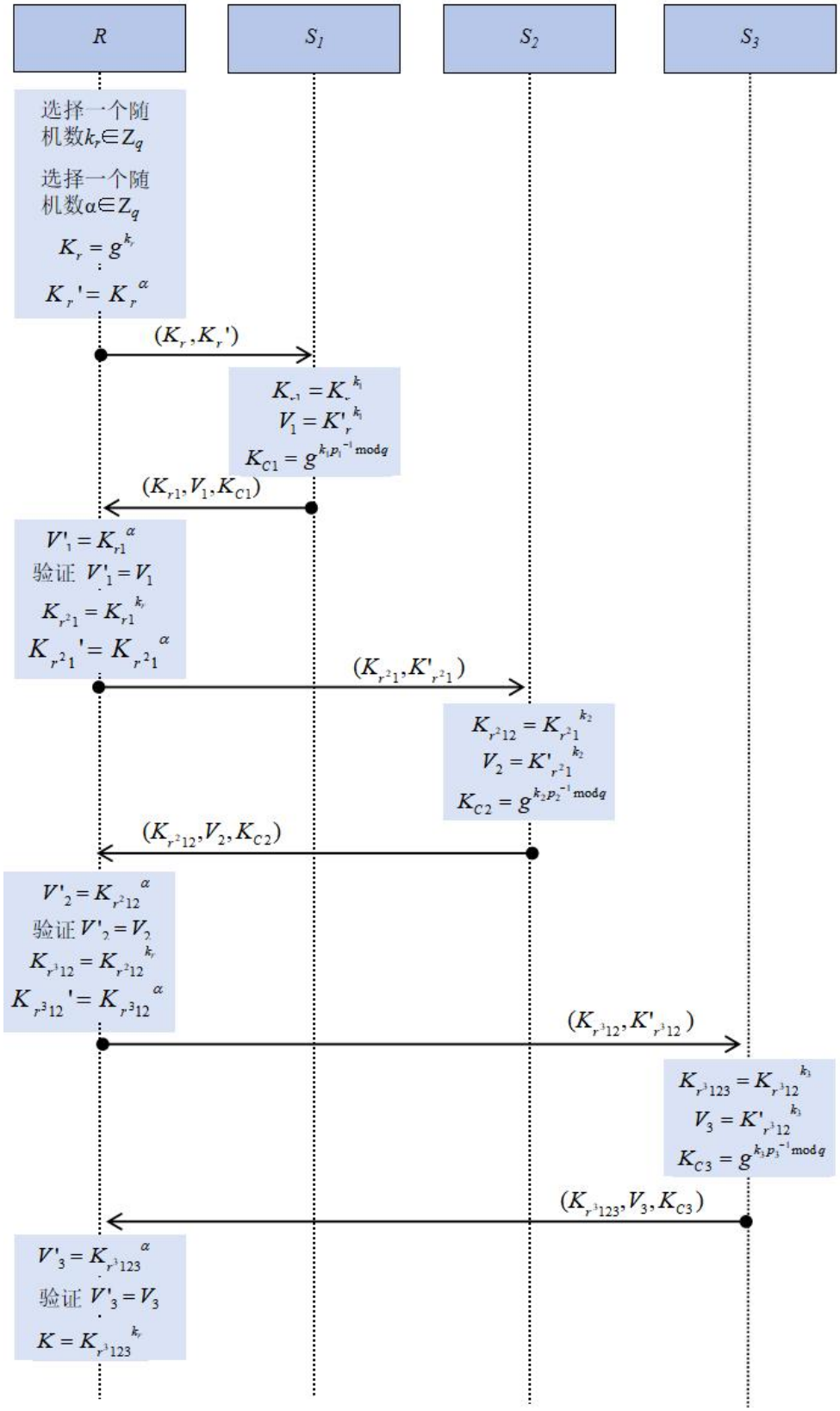


图 3-4 签名算法阶段一执行示意图

3.7.2 阶段二

签名算法阶段二的目的是继续完成签名 s。在阶段二中，接收者 R 会与签名

者们共同完成s的计算，接收者R会对签名数据进行同态加密后再发送给签名者。签名者们会对密文进行同态运算，将自己掌握的秘密随机数，掩盖因子分片，私钥分片运算到加密后的签名数据中，逐步完成被遮盖的签名分片。最后由接收者使用公开的掩盖因子去除掩盖，再聚合签名分片获得s，再最终组合成$\sigma=(r,s)$。

表 x-x 签名算法阶段二相关参数

公共参数	签名者私有参数	接收者私有参数
掩盖因子 p	私钥分片 u_i	t 个 Paillier 加密方案的公私钥
	掩盖因子分片 p_i	私钥分片的倍点 U_i
	随机数 k_i	私钥分片对应的索引 $index_i$
		随机点 K
		待签名的消息 m
		随机数 k_r

表 x-x 为签名算法阶段二相关参数信息，算法步骤如下：

1、接收者R取K的横坐标为r。

2、接收者R对要加密的消息m进行加法分片得到t份m_i，其中m_i满足:

$$m=\sum_i m_i \bmod q \tag{3-15}$$

3、接收者R使用t个 Paillier 加密方案加密r和m_i得到$c_{ri}=E_i(r)$和$c_{mi}=E_i(m_i)$。

4、接收者R选取t个随机数$\beta_i\in Z_q$，计算$c'_{ri}=\beta_i\otimes c_{ri}$和$c'_{mi}=\beta_i\otimes c_{mi}$。

5、接收者R将$(c_{ri},c_{mi},c'_{ri},c'_{mi},index_i)$发送给$S_i$。

6、签名者S_i接收到$(c_{ri},c_{mi},c'_{ri},c'_{mi},index_i)$后，根据$index_i$找到对应的$u_i$，然后计算签名分片中间结果$(c_i,c'_i)$：

$$x_i=\Gamma_i u_i \tag{3-16}$$

$$c_i=k_i^{-1}p_i\otimes[(x_i\otimes c_{ri})\oplus c_{mi}] \tag{3-17}$$

$$c'_i=k_i^{-1}p_i\otimes[(x_i\otimes c'_{ri})\oplus c'_{mi}] \tag{3-18}$$

7、签名者S_i将(c_i,c'_i)发送给接收者R。

8、接收者R接受到(c_i,c'_i)，验证下列式子是否成立，如果验证通过，则继续协议，否则中止协议。

$$\beta_i \otimes c_i = c_i' \tag{3-19}$$

9、接收者 R 计算

$$c_{Ri} = k_r^{-1} \otimes c_i \tag{3-20}$$

10、接收者 R 重新选取 t 个随机数 $\beta_j \in Z_q$，计算

$$c'_{Ri} = \beta_j \otimes c_{Ri} \tag{3-21}$$

11、接收者 R 将 (c_{Ri}, c'_{Ri}) 发送给 S_j，$S_j \in S$ 且 $S_j \neq S_i$。

12、签名者 S_j 收到 (c_{Ri}, c'_{Ri}) 后，计算：

$$c_{R^2ij} = k_j^{-1} p_j \otimes c_{Ri} \tag{3-22}$$

$$c'_{R^2ij} = k_j^{-1} p_j \otimes c'_{Ri} \tag{3-23}$$

13、签名者 S_j 将 (c_{R^2ii}, c'_{R^2ii}) 发送给接收者 R。

14、接收者 R 接收到 (c_{R^2ii}, c'_{R^2ii})。

15、接收者 R 将 (c_{R^2ii}, c'_{R^2ii}) 作为步骤 8 中的 (c_i, c'_i)，重复步骤 8~14，直到所有签名分片中间结果被所有签名者都处理了一遍。以“*”代表任意签名者的下标组合，在步骤 11 中，每次 (c_{R^*}, c'_{R^*}) 都需要发送给一个未处理过该部分签名分片的签名者。

16、完成步骤 15 接收者 R 会得到 t 份经所有签名者处理过的中间结果 (c_i, c'_i)

17、接收者 R 验证式(3-24)是否成立，如果验证通过，则继续协议，否则中止协议：

$$\beta_i \otimes c_i = c_i' \tag{3-24}$$

18、接收者 R 计算：

$$c_{si} = k_r^{-2} \otimes c_i \tag{3-25}$$

$$s_{mi} = D_i(c_{si}) \bmod q \tag{3-26}$$

$$s_m = \sum_i s_i \bmod q \tag{3-27}$$

$$s = p^{-1} s_m \bmod q \tag{3-28}$$

19、接收者 R 组合出完整的签名 $\sigma = (r, s)$

我们对阶段二的步骤做详细的解释。在阶段二中，首先步骤 1~2 由接收者 R

完成 r 的计算以及对消息 m 的分片，分片的结果 m_i 对应每个签名者 S_i。对消息分片的具体方式可以使用 Shamir 秘密共享，或者其他更方便的方法，只需要保证 $m=\sum m_i \bmod q$。

接着在步骤 3 中，接收者 R 使用不同的 Paillier 公钥加密 r 和 m_i 得到 c_{ri}和c_{mi}。在这一步中使用了不同的 Paillier 公钥加密 r 是为了得到不同的密文，以便发送给不同的签名者，保证他们不会接收到相同的信息。同时由于后续 c_{ri}和c_{mi} 需要一起进行同态运算，所以消息分片 m_i 需要与 r 在同一个公钥下加密。

在步骤 4 中，接收者 R 选取了 t 个随机数 β_i 与 c_{ri}和c_{mi} 进行同态数乘运算得到 c'_{ri} 和c'_{mi}。c'_{ri} 和c'_{mi}的作用是使得后续步骤 9 中接收者 R 可以对签名者返回的消息进行验证，确保签名者没有偏离协议。

在步骤 6 中，签名者 S_i 接收到了加密的 r 和 m_i 即 c_{ri}和c_{mi}，以及其用于校验的数据 c'_{ri}, c'_{mi}。签名者 S_i 首先通过式(3-16)将它的密钥片段 u_i 转换成加法分片 x_i，然后进行了式(3-17)的运算得到了 c_i。根据同态加密的性质 c_i 可以写作下式：

$$c_i = E_i(k_i^{-1} p_i(x_i r + m_i)) \tag{3-29}$$

从式(3-29)可以看出 c_i 是明文 $k_i^{-1} p_i(x_i r + m_i)$ 的密文。其中括号外的因子 $k_i^{-1} p_i$ 中 k_i 是当前签名者在阶段一中选择的随机数，p_i 是在初始化算法中确定的掩盖因子分片。因子 p_i 的目的是为了掩盖信息，其作用在后续步骤中将进一步说明。c_i' 与 c_i 经过了相同的运算，由于同态加密的性质 c_i' 可以写作下式：

$$c_i' = E_i(\beta_i k_i^{-1} p_i(x_i r + m_i)) \tag{3-30}$$

从式(3-30)可以看出 c_i' 对应的明文是 $\beta_i k_i^{-1} p_i(x_i r + m_i)$。它只与 c_i 的明文相差了一个因子 β_i。

步骤 8 中接收者 R 对 c_i 和 c_i' 进行验证计算，验证 c_i' 是否由 β_i 同态数乘 c_i 所得。如果验证不通过说明签名者 S_i 没有依照协议对 c_{ri} 和 c_{mi} 做指定的运算，此时中止当前协议。如果检查通过了则继续进行协议。

步骤 9，接收者 R 使用 k_r^{-1} 与 c_i 作同态乘法得到 c_{Ri}。这个步骤有两个作用，一方面进一步完成了签名分片的计算，此时 c_{Ri} 的明文为 $(k_r k_i)^{-1} p_i(x_i r + m_i)$；另一方面也避免了将上一个签名者的数据直接发送给下一个签名者的情况。

步骤 10 的作用与步骤 4 类似。

步骤 11 将 (c_{Ri}, c'_{Ri}) 发送给下一个从未处理过当前签名的分片数据的签名者。

我们举个例子来更好地说明此步选取下一个签名者的要求，假设一个签名分片数据c_{R1}，它的值如下：

$$c_{R1} = E_1(k_r^{\ 3}k_1k_2)^{-1}p_1p_2(m_1 + rx_1) \tag{3-30}$$

对应的明文是$(k_r^{\ 3}k_1k_2)^{-1}p_1p_2(m_1 + rx_1)$，从左边的因子$(k_r^{\ 3}k_1k_2)^{-1}p_1p_2$可以看出它已经被$S_1$和$S_2$处理过了，因为这个数据都包含了这两个签名者选取的随机值k_i以及掩盖因子分片p_i。因此这个c_{R1}不需要再发给S_1或S_2进行处理，只能发给除他们以外的任意一个签名者，比如S_3。接收者R可以根据每次数据交互的对象来记录每个签名分片数据已经被哪些签名者处理以及还需要发送给哪些签名者。

步骤 12~13，签名者使用同态运算将$k_j^{\ -1}p_j$乘入c_{Ri}和c'_{Ri}后回发给接收者R。在这一步中，乘入$k_j^{\ -1}$是为了完成签名，而p_j的作用是为了掩盖$k_j^{\ -1}$。p_j的作用具体而言可以从接收者R视角进行分析，在步骤 13 中，接收者R发送c_{Ri}给签名者S_j，在步骤 16 中，接收者接收到了来自S_j处理过的c_{Rij}。此时接收者R可以对c_{Ri}和c_{Rij}进行解密获得明文，假设c_{Ri}的明文为$(k_rk_i)^{-1}p_i(x_ir + m_i)$,那么$c_{Rij}$的明文为$(k_rk_ik_j)^{-1}p_ip_j(x_ir + m_i)$,接收者$R$对两个明文进行以下运算：

$$(k_rk_ik_j)^{-1}p_ip_j(x_ir + m_i)[(k_rk_i)^{-1}p_i(x_ir + m_i)]^{-1} \bmod q = k_j^{\ -1}p_j \bmod q \tag{3-31}$$

而$k_j^{\ -1}p_j$正是S_j乘入c_{Ri}的数值，假如没有p_j这个因子，那么接收者R可以直接分析c_{Ri}和c_{Rij}得知S_j的秘密值k_j，这样是不安全的。因此需要引入一个掩盖因子p_j，在接收者R未知$k_j^{\ -1}$和p_j的情况下，从$k_j^{\ -1}p_j$计算出k_j是相对困难的，由此保证签名者选取的秘密随机值不会泄露。

步骤 17 的作用类似步骤 8。

步骤 18，接收者R已经获得了所有签名分片数据c_i，此时这些签名分片数据的明文是

$$D_i(c_i) = (k_r^{\ t-1}\prod_i k_i)^{-1}(\prod_i p_i)(m_i + x_ir) = (k_r^{\ t-1}\prod_i k_i)^{-1}p(m_i + x_ir) \tag{3-32}$$

与真正的签名分片s_i还差$(k_r^{\ 2})^{-1}$并且还需要去除掩盖因子p。式(3-25)的作用是将$(k_r^{\ 2})^{-1}$运算到签名中，因此有：

$$s_{mi} = D_i(c_{si}) = (k_r^{\ t+1}\prod_i k_i)^{-1}p(m_i + x_ir) \tag{3-33}$$

此时得到的s_{mi}是依旧被掩盖因子遮盖的签名分片。将s_{mi}求和得到被掩盖的

签名 s_m：

$$s_m = \sum_i s_{mi} = (k_r^{t+1} \prod_i k_i)^{-1} p(\sum_i m_i + \sum_i x_i r) = (k_r^{t+1} \prod_i k_i)^{-1} p(m + xr) \quad (3\text{-}34)$$

式(3-28)是使用公开的掩盖因子 p 去除掩盖因子得到 s，展开如下式

$$s = p^{-1} s_m = (k_r^{t+1} \prod_i k_i)^{-1} (m + xr) \quad (3\text{-}35)$$

再结合式(3-14)则可将 s 化为：

$$s = k^{-1}(m + xr) \quad (3\text{-}26)$$

式(3-26)形式与 ECDSA 数字签名方案中 s 一致，说明最终结果是一个有效的 ECDSA 数字签名。以接收者与三个签名者完成签名算法为例，图 3-4 和图 3-5 是一次签名算法阶段二的执行示意图。

3.7.3 签名验证

TB-ECDSA 算法生成的数字签名与 ECDSA 的签名形式是一致的，因此验签方式与 ECDSA 一致，接收者 R 只需要计算

$$K' = g^{ms^{-1}} y^{rs^{-1}} \quad (3\text{-}27)$$

取 K' 的横坐标位 r'，验证 $r' = r$ 是否成立。如果验证通过，说明生成了合法的签名。如果验证不通过，说明签名者中有人偏离了协议，造成了最终验证失败。

3.8 错误识别

当接收者发现生成的签名无法正常验签时，可以通过签名算法中的中间数据来排查出偏离协议导致签名失效的签名者。TB-ECDSA 方案可以排查出以下两类偏离协议的签名者：

1、第一类偏离协议：如果签名者 S_i 在签名算法阶段二步骤 7 中偏离了协议，发生了以下几种情况之一：

（1）没有使用正确的私钥分片 u_i 签名。

（2）没有使用正确的掩盖因子分片 p_i。

（3）没有使用阶段一所选取的随机数 k_i。

则接收者 R 可以通过以下步骤排查出该签名者：

计算：

$$s_{i\_temp} = D_i(c_i) \tag{3-38}$$

验证下式是否成立。

$$U_i^{\Gamma_i r} g^{m_i} = K_{C1}^{s_{i\_temp}} \tag{3-39}$$

假如成立，说明S_i在步骤 7 没有出现偏离协议，否则说明S_i出现了第一类偏离协议。

我们对验证的原理进行解释，对于第一类偏离协议的签名者的排查，假如签名者没有偏离协议，s_{i_temp}应该有如下取值：

$$s_{i\_temp} = k_i^{-1} p_i (m_i + x_i r) \tag{3-40}$$

我们对式(3-39)等号两边分别逐级展开如式(3-41)和式(3-42)。

$$U_i^{\Gamma_i r} g^{m_i} = g^{\Gamma_i u_i r} g^{m_i} = g^{m_i + x_i r} \tag{3-41}$$

$$K_{C1}^{s_{i\_temp}} = g^{k_i p_i^{-1} s_{i\_temp}} = g^{k_i p_i^{-1} k_i^{-1} p_i (m_i + x_i r)} = g^{m_i + x_i r} \tag{3-42}$$

可以看出当签名者遵循协议流程时，式(3-39)左右两边应该是相等的。而当签名者偏离了协议时会导致$K_{C1}^{s_{i_temp}} \neq g^{m_i + x_i r}$，使得式(3-39)不成立。

2、第二类偏移协议：如果签名者S_i在签名算法阶段二步骤 14 中偏离了协议，发生了以下几种情况之一：

（1）没有使用正确的掩盖因子分片p_i。

（2）没有使用阶段一所选取的随机数k_i。

（3）则接收者R可以通过以下步骤排查出该签名者。

计算：

$$s_{i\_temp} = D_i(c_{Rij}) \tag{3-43}$$

$$s_{i\_temp2} = D_i(c_{Ri}) \tag{3-44}$$

验证下式是否成立：

$$K_{Ci}^{s_{i\_temp}} = g^{s_{i\_temp2}} \tag{3-45}$$

假如成立，说明S_i在步骤 14 中没有出现偏离协议，否则说明S_i出现了第二类偏离协议。

我们对验证的原理进行解释，对于第二类偏离协议的签名者的排查，假如签名者没有偏离协议，s_{i_temp}和s_{i_temp2}应该有如下取值：

$$s_{i\_temp} = k_j^{-1} p_j k_i^{-1} p_i (m_i + x_i r) \tag{3-46}$$

$$s_{i\_temp2} = k_i^{-1} p_i (m_i + x_i r) \tag{3-47}$$

我们对式(3-45)等号两边分别逐级展开：

$$K_{Cj}^{s_{i\_temp}} = g^{k_j p_j^{-1} k_j^{-1} p_j k_i^{-1} p_j (m_i + x_i r)} = g^{k_i^{-1} p_i (m_i + x_i r)} \tag{3-48}$$

$$g^{s_{i\_temp2}} = g^{k_i^{-1} p_i (m_i + x_i r)} \tag{3-49}$$

可以看出当签名者遵循协议流程时，式(3-45)左右两边应该是相等的。而当签名者偏离了协议时会导致 $K_{Cj}^{s_{i\_temp}} \neq g^{k_i^{-1} p_i (m_i + x_i r)}$，使得式(3-45)不成立。

3.9 安全性分析

3.9.1 不可伪造性

假设有一个攻击者 A，他已知 l 个消息 $(m_1, m_2, ..., m_l)$，以及这些消息对应的签名 $(\sigma_1, \sigma_2, ..., \sigma_l)$。假如 A 无法根据 $(m_1, m_2, ..., m_l)$ 和 $(\sigma_1, \sigma_2, ..., \sigma_l)$ 对一个未签名过的消息 m_{l+1} 生成签名 σ_{l+1}，则称这个签名方案是不可伪造的。

TB-ECDSA 是满足不可伪造性的。TB-ECDSA 数字签名等价于一个 ECDSA 数字签名算法计算出的签名。攻击者 A 获取到 l 个 TB-ECDSA 算法计算出的数字签名相当于获取到 l 个 ECDSA 数字签名。而 ECDSA 数字签名具有不可伪造性，因此攻击者无法从 l 个 TB-ECDSA 数字签名中生成一个新的有效的签名。因此 TB-ECDSA 具有不可伪造性。

3.9.2 门限特性

如果在一个签名方案中，私钥 x 以分片的形式保存在 n 个对象手中，完成一次合法的签名需要其中的 t 个对象参与，如果不足 t 个对象参与签名，则无法完成合法签名，称这样的签名方案是满足门限特性的。

TB-ECDSA 是满足门限特性的。在 TB-ECDSA 接收者初始化算法中，接收者 R 以私钥 x 为秘密构造了 Shamir 秘密共享，将私钥分片 u_i 分发给了 n 个签名者。根据 Shamir 秘密共享的特性，只有当 n 位签名者中的 t 位提供他们的私钥分片才可以将私钥 x 还原出来。根据 Shamir 秘密共享的特性，假如不足 t 位签名者提供私钥分片，则无法还原私钥 x，进而也不可能完成一个合法的 ECDSA 签名。综上 TB-ECDSA 是满足门限特性的。

3.9.3 盲性与不可链接性

盲性与不可链接性指的是在签名过程中，签名者无法获取与签名相关的信息，也无法确定签名过程与后续公开签名之间的确切关系。

TB-ECDSA 签名算法是满足盲性与不可链接性的。在签名算法的阶段一，假设接受者与签名者交互的顺序是 $S_1,S_2,...,S_t$ 则在阶段一中，接收者分别向签名者 S_i 揭露了 $g^{k_r{}^i\prod_i^{k_{i-1}}}$，而与签名相关的信息是 k 和 K，其值分别是：

$$k = k_r{}^{t+1}\prod_{0<i<t} k_i \tag{3-50}$$

$$K = g^{k_r{}^{t+1}\prod_{0<i<t} k_i} \tag{3-51}$$

而在不知道 k_r 的情况下，每个 S_i 无法由 $g^{k_r{}^i\prod_i^{k_{i-1}}}$ 推导得到 k 和 K。因此在阶段一过程中，没有签名者得知了 k 和 K 的具体值。假如签名者试图篡改数据来获取更多信息，也会因为其偏离了协议在阶段一中的验证步骤或者在最后进行错误识别时被检查出来。

在阶段二中，接收者首先对消息 m 进行分片，再对其加密得到 c_{mi}，发送给签名者 c_{mi} 和 c_{ri}，这些数据都是经过同态加密的。假如 Paillier 加密方案是语义安全的话，在没有对应的解密密钥的情况下，签名者是将这些数据与随机数区分开的，因此他们无法从这些数据中得知任何关于签名的敏感信息。在阶段二的后续步骤中签名者接收的都是经过其他签名者以及接收者同态运算过的加密的数据，这些数据同理也是无法被签名者解密或者提取更多有用的信息的。假如签名者试图通过篡改数据的方法来获取更多信息，也会因为其偏离了协议，在阶段二中步骤 9 的验证或者最后进行错误识别时被检查出来。

因此签名者在签名的整个过程中都无法得知关于签名的任何相关的信息，那么他也无法确定签名过程与后续公开签名之间的确切关系。

3.9.4 用户匿名性

我们分别从 TB-ECDSA 签名算法的两个阶段说明该算法具有用户匿名性，其中签名算法阶段一的部分采用较为严谨的证明的方式，签名算法二的部分采用直观地说明的方式。

1、证明 TB-ECDSA 签名算法阶段一满足用户匿名性：

命题 3-1 如果判定性 DDH 假设成立，那么 TB-ECDSA 签名算法阶段一是

满足用户匿名性的。

上述命题的逆否命题为：如果 TB-ECDSA 签名算法阶段一是不满足用户匿名性的，那么决定性 DDH 假设不成立。对该逆否命题的证明如下：

假设有接收者 R_1 和 R_2，签名者 S_1 和 S_2。R_1 和 R_2 分别与签名者们完成一次阶段一的算法。R_1 和 R_2 选择的秘密随机数分别为 k_{r1} 和 k_{r2}。R_1 和 R_2 选择的另一随机数为 α_1 和 α_2。S_1 在响应 R_1 和 R_2 的过程中选择的随机数分别为 k_{11} 和 k_{12}。此时 S_1，S_2 收到的所有数据如下：

$$S_1:((g^{k_{r1}},g^{\alpha_1 k_{r1}}),(g^{k_{r2}},g^{\alpha_2 k_{r2}})) \tag{3-52}$$

$$S_2:((g^{{k_{r1}}^2 k_{11}},g^{\alpha {k_{r1}}^2 k_{11}}),(g^{{k_{r2}}^2 k_{12}},g^{\alpha {k_{r2}}^2 k_{12}})) \tag{3-53}$$

因为 TB-ECDSA 中的阶段一是不满足用户匿名性的，则存在一个攻击者 A 控制了 S_1，S_2 得知了他们收到的所有数据，A 有能力通过多项式时间内的函数 F 以不可忽略的优势确定两个数据是否来自同一个接收者，如式(3-54)所示，当 F 输出 1 时表示输入的两个循环群元素来自同一个接收者，如式(3-55)当 F 输出 0 时表示输入的两个群元素来自不同的接收者：

$$F(g^{k_{r1}},g^{{k_{r1}}^2 k_{11} \bmod q})=1 \tag{3-54}$$

$$F(g^{k_{r1}},g^{{k_{r2}}^2 k_{12} \bmod q})=0 \tag{3-55}$$

一个等价于判定性 DDH 假设的命题如下：

给定阶为 q 的循环群 G，其中 q 是一个大素数，g 为循环群 G 的生成元，且对于任意的 $a,b,c\in Z_q$，总能找到 $d=ba^{-1}$。无法在多项式时间内以不可忽略的优势区分以下两个四元组：

$$R=\left(g,g^a,g^{ad},g^c\right) \tag{3-56}$$

$$D=\left(g,g^a,g^{ad},g^{a^2 d}\right) \tag{3-57}$$

利用 A 的能力 F 可以在多项式时间内以不可忽略的优势区分 (g,g^a,g^{ad},g^c) 和 $(g,g^a,g^{ad},g^{a^2 d})$ 如下：

$$F(g^a,g^{a^2 d})=1 \tag{3-58}$$

$$F(g^a,g^c)=0 \tag{3-59}$$

因此判定性 DDH 假设不成立。

所以命题 3-1 的逆否命题为真，那么命题 3-1 也为真。

2、TB-ECDSA 签名算法阶段二满足用户匿名性：

假设有接收者 R_1 和 R_2，签名者 S_1, S_2 和 S_3。R_1 和 R_2 分别与签名者们完成一次阶段二的算法。此时 S_1, S_2 和 S_3 接收到的部分数据如下：

$$S_1:((c_{r_11}, c_{m_11}, c'_{r_11}, c'_{m_11}), (c_{r_21}, c_{m_21}, c'_{r_21}, c'_{m_21})) \tag{3-60}$$

$$S_2:((c_{R_11}, c'_{R_11}), (c_{R_21}, c'_{R_21})) \tag{3-61}$$

$$S_3:((c_{R_1^2 12}, c'_{R_1 12}), (c_{R_2^2 12}, c'_{R_2 12})) \tag{3-62}$$

假设一个攻击者 A 控制了 S_1, S_2 和 S_3，那么他得到了以上数据。由于 Paillier 加密方案是语义安全的，在 A 没有获得对应的私钥的情况下，他无法从这些数据中提取有效的信息，因此它无法无法根据数据的内容对这些数据按接收者进行分类。例如，A 无法判断数据 $(c_{r_11}, c_{m_11}, c'_{r_11}, c'_{m_11})$ 与 (c_{R_11}, c'_{R_11})和(c_{R_21}, c'_{R_21})中的哪一个来自同一个接收者。同理对于其他的数据组合也是一样的。所以 TB-ECDSA 签名算法阶段二满足用户匿名性。

综上所述，签名算法的阶段一与阶段二皆满足用户匿名性，因此签名算法满足用户匿名性。

3.10 计算性能分析

m_p 代表一次椭圆曲线的倍点运算，exp 代表一次模指数运算，$|x|$代表一个整数 x 的比特长度，t 代表参与运算的签名者数量。因为模加运算和模乘运算的运算资源消耗相对于椭圆曲线倍点运算和模指数运算是比较小的，因此我们没有统计 TB-ECDSA 中的这两类运算。

表 3-1 统计了在签名算法阶段一的计算和通信复杂度，其中 q 为公共参数椭圆曲线群的阶，t 为参与的签名者数量。并且，由于步骤 7 是重复步骤 3~6，因此步骤 7 的统计为步骤 7 以前的计算复杂度的累加再乘以重复的次数，重复的次数取决于参与的签名者数量，在此处为 $(t-1)$。

表 3-1 签名算法阶段一的计算和通信复杂度

步骤	接收者计算复杂度	签名者计算复杂度	通信复杂度(bit)
1~4	2mp		4|q|
5		3mp	6|q|
6	mp		

7	(t-1)3mp	(t-1)3mp	10(t-1)|q|
8~9	mp		
总计	(3t+1)mp	3tmp	10t|q|

表 x-x 统计了签名算法阶段二的计算和通信复杂度，其中 *index* 为私钥索引，N 为 Paillier 公钥，由于在阶段二中用到了多个 Paillier 公钥，因此$|N^2|$表示的是多个 Paillier 加密密文构成的平均值。步骤 15 是重复步骤 8~14，因此步骤 15 的统计为步骤 8~14 的计算复杂度乘以重复的次数，重复的次数取决于参与的签名者数量，在此处为$(t-2)$。另外虽然 Paillier 加密的计算是$c=g^m r^N \bmod N^2$，但是因为$g=(1+N)^{pt} \bmod N^2$，所以$g^m=(1+mptN) \bmod N^2$，所以 Paillier 加密计算就可以等效为$c=(1+mptN)r^N \bmod N^2$。可以看出 Paillier 加密计算只需要进行一次模指数运算。因此统计时对于每次 Paillier 加密都当作是一次模指数运算。

表 3-2 签名算法阶段二的计算和通信复杂度

步骤	接收者计算	签名者计算	通信复杂度(bit)
1~5	4t exp		t(|N2|+|index|)
6~7		4t exp	2t|N2|
8~14	3t exp	2t exp	4t|N2|
15	(t-2)3t exp	(t-2)2t exp	(t-2)4t|N2|
17~19	3t exp		
总计	(3t2+4t) exp	(2t2+2t)exp	(4t2-t)|N2|+t|index|

综合表 3-1 和表 3-2，我们可以得出 TB-ECDSA 签名算法接收者计算复杂度为：

$$(3t+1)m_p+(3t^2+4t)\exp \tag{3-63}$$

签名者的计算复杂度为：

$$3tm_p+(2t^2+2t)\exp \tag{3-64}$$

通信复杂度为：

$$10t|q|+(4t^2-t)|N^2|+t|index| \tag{3-65}$$

3.11 本章小结

在本章中，我们在用户匿名性的规则限制下，构建了一个门限盲签名方案，

其由接收者和签名者组成。我们首先设计了签名者初始化算法，采用了本章提出的 MSP 算法。接着，我们利用 Shamir 秘密共享技术完成了接收者的密钥分配，进而完成了接收者初始化算法的设计。随后，我们利用 Paillier 同态加密的特性完成了签名算法的设计。我们还详细介绍了该方案中的错误识别特性，即接收者可以通过分析算法的中间数据，准确定位导致签名失败的签名者。

最后，我们对提出的 TB-ECDSA 方案进行了安全性分析，阐明了其具备的几个特性，并从理论角度对该方案的计算性能进行了深入分析。

第四章 基于多方计算的 ECDSA 门限盲签名系统实现

4.1 开发环境搭建

表 4-1 系统实现软硬件参数表

名称	配置参数
操作系统	Windwos 10
CPU	Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz
内存	16G
硬盘	500G
数据传输格式	Protocol Buffer

表 4-1 为本系统实现的软硬件参数表，在本研究中，我们选择 Go 语言作为开发工具，并以 Visual Studio Code (VSCode)作为集成开发环境，以便高效地进行代码编写。系统中涉及对象间的数据交换，我们采用了 Protocol Buffer（ProtoBuf）构造数据格式。另外，我们基于开源的密码学库 tss-lib 构建了本系统，该库提供了实现门限签名方案所需的密码学基础功能以及多方计算的系统框架，保障了系统的安全性和可靠性。

4.1.1 Go

Go 语言是一种由 Google 开发的开源编程语言，其突出的特性之一就是对并发的支持。Go 语言内置了轻量级的协程，使得并发编程变得简单且高效。通过使用 go 关键字启动一个新的协程，可以轻松地实现并发操作，而不需要显式地管理线程。

4.1.2 Protocol Buffer

我们采用了 ProtoBuf 作为数据交换格式。ProtoBuf 是一种由 Google 开发的轻量级数据交换格式，旨在提供高效的序列化机制。其设计的目标包括跨平台、跨语言的支持、自描述性和可扩展性。

通过使用 ProtoBuf，我们能够定义数据结构和消息格式，并生成相应的代码

用于数据的序列化和反序列化。这种方式不仅能够减少数据传输时的开销，还能提高数据传输的效率和可靠性。

使用 ProtoBuf 需要首先需要使用 ProtoBuf 语法编写 proto 文件来定义消息类型。如图 4-1 种代码所示，这段 ProtoBuf 代码定义了一种消息类型，消息名称为 SignRound2Message2，其中包含四个字段，名称分别为 Ci、Ci_a、bigXi_x 以及 bigXi_y。

```
message SignRound2Message2{
    bytes Ci = 1;
    bytes Ci_a = 2;
    bytes bigXi_x = 3;
    bytes bigXi_y = 4;
}
```

图 4-1 使用 ProtoBuf 语法定义消息类型示例

编写完 proto 文件后通过命令行或者相关插件，调用 protoc.exe 对指定文件进行转译。ProtoBuf 会根据 proto 文件生成 Go 代码，代码中会包含数据的序列化和反序列化方法。图 4-2 是图 4-1 中定义的消息经过 ProtoBuf 后生成的部分 Go 代码。

```
type SignRound2Message2 struct {
    state         protoimpl.MessageState
    sizeCache     protoimpl.SizeCache
    unknownFields protoimpl.UnknownFields

    Ci     []byte `protobuf:"bytes,1,opt,name=Ci,proto3" json:"Ci,omitempty"`
    CiA    []byte `protobuf:"bytes,2,opt,name=Ci_a,json=CiA,proto3" json:"Ci_a,omitempty"`
    BigXiX []byte `protobuf:"bytes,3,opt,name=bigXi_x,json=bigXiX,proto3" json:"bigXi_x,omitempty"`
    BigXiY []byte `protobuf:"bytes,4,opt,name=bigXi_y,json=bigXiY,proto3" json:"bigXi_y,omitempty"`
}

func (x *SignRound2Message2) Reset() {
    *x = SignRound2Message2{}
    if protoimpl.UnsafeEnabled {
        mi := &file_protob_ecdsa_blind_signing_proto_msgTypes[3]
        ms := protoimpl.X.MessageStateOf(protoimpl.Pointer(x))
        ms.StoreMessageInfo(mi)
    }
}

func (x *SignRound2Message2) String() string {
    return protoimpl.X.MessageStringOf(x)
}
```

图 4-2 使用 ProtoBuf 生成的部分代码

4.1.3 tss-lib

tss-lib 是一个开源的密码学库，由 Binance Smart Chain (BSC)团队开发和维护。它提供了实现门限签名方案所需的密码学基础功能以及一个多方计算的系统框架。

其中 tss-lib 库中的 crypto 包提供了实现 TB-ECDSA 需要的椭圆曲线密码学算法，包括倍点运算、标量乘法等操作。

表 4-1　crypto.ScalarBaseMult 函数各参数的意义

包名	crypto
函数名	ScalarBaseMult
输入	
名称：类型	含义
curve ： elliptic.Curve	椭圆曲线的参数
k ： *big.Int	一个大整数，用作倍点的乘法因子
输出	
类型	含义
*ECPoint	椭圆曲线基点作 k 倍点运算的结果

如表 4-1 所示，调用 crypto.ScalarBaseMult 函数可以完成一次椭圆曲线基点的倍点运算，它接受一个椭圆曲线 curve 和一个大整数 k 作为参数，并返回一个椭圆曲线上的点的结果。

表 4-2　ECPoint.ScalarMul 函数各参数的意义

包名	crypto
函数名	ECPoint.ScalarMul
输入	
名称：类型	含义
k ： *big.Int	一个大整数，用作数量的乘法因子
输出	
类型	含义
*ECPoint	该点经过 k 倍点运算的结果

如表 4-2 所示，调用 ECPoint.ScalarMul 函数可以完成一次任意椭圆曲线点的倍点运算，它接收一个大整数 k 作为参数，并返回椭圆曲线上的一个点，该点是调用 ScalarMul 的椭圆曲线点与 k 倍点运算的结果。

tss-lib 库中的 common 包提供了获取随机数还有大素数的程序调用接口，调用相关函数可以方便地生成随机数和大素数。

表 4-3　common.GetRandomPositiveInt 函数各参数的意义

包名	common
函数名	GetRandomPositiveInt
输入	
名称：类型	含义
Rand：io.Reader	一个随机数生成器，用于生成随机数
lessThan：*big.Int	生成随机数的上限
输出	
类型	含义
*big.Int	一个小于 lessThan 的随机正整数

如表 4-3 所示，调用 common.GetRandomPositiveInt 函数可以生成一个小于给定大整数 lessThan 的随机正整数。它接收一个随机数生成器 rand 和一个上限值 lessThan 作为参数，并返回一个小于 lessThan 的随机正整数。

表 4-4　common.GetRandomSafePrimesConcurrent 函数各参数的意义

包名	common
函数名	GetRandomSafePrimesConcurrent
输入	
名称：类型	含义
Ctx：context.Context	上下文对象
bitLen：int	生成的安全素数的比特位长度
numPrimes：int	要生成的安全素数的数量
Concurrency：int	并发级别，控制同时进行的生成任务数量
rand io：Reader	随机数生成器，用于生成随机数
输出	
类型	含义
[]*GermainSafePrime	安全素数组成的数组

如表 4-4 所示，调用 common.GetRandomSafePrimesConcurrent 可以并发生成安全素数。通过使用多个核心并行地生成安全素数，函数可以更快地找到多个有效的结果。

4.2 系统功能实现

4.2.1 系统框架

图 4-3 展示了本系统框架在完成一次消息发送/接收时的情况。我们以参与者 A 向参与者 B 发送一次消息为例，阐述本系统框架的构成以及工作过程。

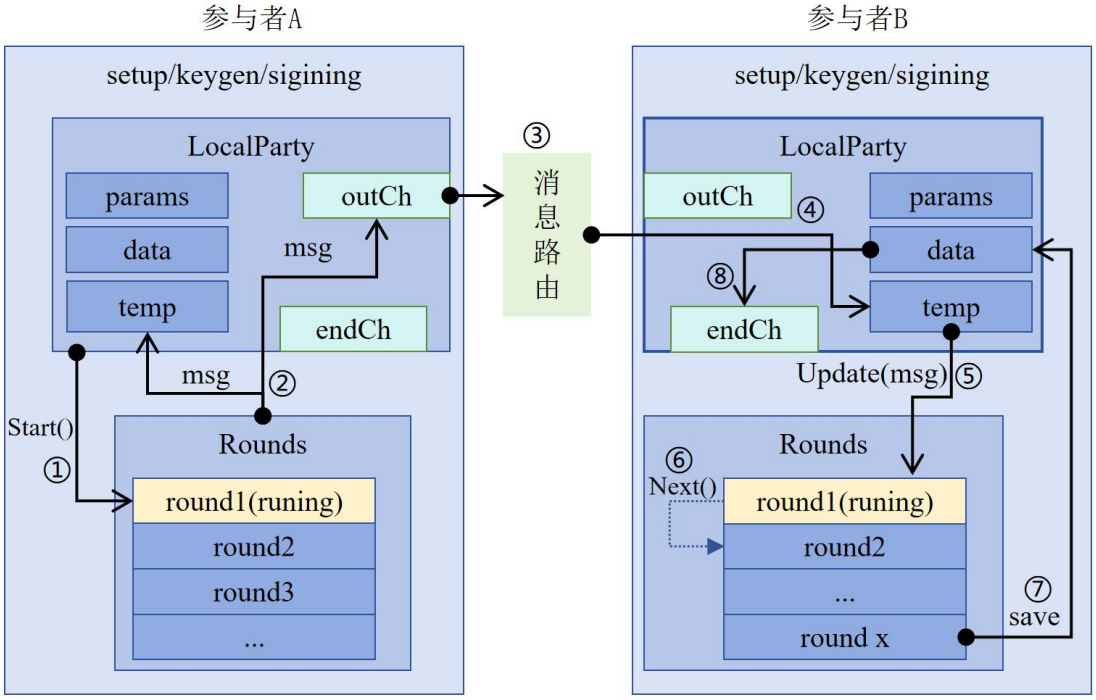


图 4-3 系统框架示意图

在本系统的实现中，我们定义了三个包对应 TB-ECDSA 方案的三个算法：

1、setup 包：签名者初始化算法。

2、keygen 包: 接收者初始化算法。

3、signing 包：签名者初始化算法。

每个包中，都定义了两种重要的对象：LocalParty 和 Rounds，它们在不同的包中具体会有不用的定义，但总的来说 LocalParty 会有以下五类数据结构：

1．params：当前 LocalParty 的一些参数，包括采用的椭圆曲线，算法的参与方信息，当前配置的门限值等等。

2．data: 当前 LocalParty 持有的重要数据，在算法开始时从外部读取，在算法结束时持久化储存，包括一些公共参数，私有数据，签名结果等等。

3．temp：在进行算法时需要保存的临时数据，包括所有收到的消息，中间变量或是需要在后续步骤用到的数据等。

4．outCh: 用于发送传出消息的通道。

5．endCh：用于返回计算完成的 data 的通道。

Rounds 类型中包含了当前算法每个轮次需要运行的算法步骤，分别写在 round1、round2 等结构中。

在图 4-3 中一个轮次会这样进行:

1、当参与者 A 完成了 LocalParty 的初始化后，会调用 Start()来运行第一轮 round1 的算法。

2、当 round1 的算法运行完成后，会将要发出的消息 msg 保存到 temp 中，然后再将 msg 通过 outCh 发送出去。

3、消息经过外部的消息路由。

4、消息到达指定的其他 LocalParty 的 temp 中，在图 4-3 中接收 msg 的是参与者 B。

5、参与者 B 接收到 msg 后，会将 msg 从 temp 中取出调用 Update(msg)来把消息更新到 round1 中。

6、round1 收到 msg 后，就会调用 Next()，来进行下一轮的运算。

7、当参与者 B 完成了当前算法的所有轮次运算后，就会将结果保存到 data

8、接着 data 通过 endCh 传出。

4.2.2 签名者初始化模块

根据第三章 3.3 小节的算法，我们将签名者初始化分为三个轮次。

表 x-x 签名者初始化算法第一轮次伪代码

签名者初始化算法：Round1	
1.	pi:=common.GetRandomSafePrimesConcurrent()
2.	prime_share[] := CreateShamirShare(pi,Signers[])
3.	for i=0;i<Len(Signers);i++{
4.	Send(prime_share[i],Signers[i])
5.	}
6.	while(true){
7.	if(Listen()!=null)
8.	break
9.	}

```
10.  NextRound()
```

第一轮次如表 x-x 伪代码所示，签名者选取掩盖因子分片，并进行 Shamir 秘密共享的相关计算，将相应的数据发送到对应的节点，然后开始监听消息，辅助签名者在第一轮直接进行消息监听。当各个签名者收到来自其他签名者的消息时，进入到下一轮次。

表 x-x 签名者初始化算法第二轮次伪代码

```
签名者初始化算法：Round2
1.  Shares[] := Listen()
2.  gama_share :=1
3.  for i:=0;i<Len(Shares);i++{
4.          gama_share = gama_share*Shares[i]
5.  }
6.  gama_share = ConvertToAddingShare(gama_share)
7.  Boradcast(gama_share )
8.  NextRound()
```

第二轮次如表 x-x 伪代码所示，各个签名者按照算法对接受到的数据进行处理，得到掩盖因子加法分片，然后广播该数据并进入消息监听状态。当各个签名者接收到了其他签名者的广播后，进入到下一轮次。

表 x-x 签名者初始化算法第三轮次伪代码

```
签名者初始化算法：Round3
1.  gama_shares[] := Listen()
2.  prime_mask:=Sum(gama_shares)
3.  Save(prime_mask)
4.  Boradcast(prime_mask)
5.  End()
```

第三轮次如表 x-x 伪代码所示，从消息中提取掩盖因子加法分片，计算出掩盖因子，将掩盖因子持久化储存后广播，最后结束算法。

如表 4-5 根据这三个轮次的通讯的消息，我们可以定义三种消息类型，分别对应三个轮次中发送的消息。

表 4-5 签名者初始化中的消息类型

消息名称	字段名称	类型	含义	对应符号
SURound1Message	prime_share	bytes	掩盖因子分片的分片	p_i
SURound2Message	gama_share	bytes	掩盖因子加法分片	p_{mi}
SURound3Message	prime_mask	bytes	掩盖因子	p

4.2.3 接收者初始化模块

根据第三章 3.4 小节我们将接收者初始化算法分为三个轮次。

表 x-x 接收者初始化算法第一轮次伪代码

接收者初始化算法：Round1	

接收者：
Private_key:=GeneratePrivateKey()
shares[]:=Shamir(Private_key,Signers[])
for i:=0;i<Len(Signers);i++{
　　Send(shares[i], Signers[i])
}
NextRound()
	签名者：
	if(Listen() != null){
	　　NextRound()
	}

第一轮次如表 x-x 伪代码所示，签名者直接进入监听状态，接收者生成私钥，并对私钥进行 Shamir 秘密分享的计算，将产生的私钥分片发给对应的签名者，然后进入第二轮次。当签名者收到接收者的消息时进入第二轮次。

表 x-x 接收者初始化算法第二轮次伪代码

接收者初始化算法：Round2	

接收者：
if(Listen()!= null){
　　NextRound()
}
	签名者：
	Share := Listen()
	Index := Save(share)
	Send(index, Recipient)
	End()

第二轮次如表 x-x 伪代码所示，签名者从消息中提取出私钥分片，在本地持久化保存，并将私钥索引发回给接收者。接收者在第二轮次直接进入监听状态，等待签名者的消息。当接收者收到签名者发来的消息时，进入下一轮次。

表 x-x 接收者初始化算法第三轮次伪代码

接收者初始化算法：Round3

接收者：	签名者：
Index := Listen()	无需行动
Save(index)	
End()	

第三轮如表 x-x 伪代码所示，签名者无需行动。接收者从消息中提取出索引，并在本地持久化保存，结束算法。

如表 4-6，根据这三个轮次的通讯的消息，我们可以定义两种消息类型。

表 4-6 接收者初始化中的消息类型

消息名称	字段名称	类型	含义	对应符号
KGRound1Message	share	bytes	私钥分片	u_i
KGRound2Message	index	bytes	私钥索引	$index_i$

4.2.4 签名算法模块

根据第三章 3.5 小节，我们可以将签名算法分为四个轮次，其中有两个轮次是需要执行多次。

表 x-x 签名算法第一轮次伪代码

签名算法：Round1	
接收者：	签名者：
BigKr:=crypto.ScalarBaseMult(kr)	while(Listen() == null){
BigKr_a := BigKr.ScalarMul(alpha)	Wait()
for i:=0;i<Len(Signers);i++{	}
Send((BigKr,BigKr_a), Signers[i])	BigKr, := Listen()
While(Listen==null){	BigKr_a:=Listen()
Wait()	ki:=common.GetRandomPositiveInt()
}	BigKri := BigKr.ScalarMul(ki)
BigKri := Listen()	BigVi := BigKr.ScalarMul(ki)
BigVi:=Listen()	BigKci:=crypto.ScalarBaseMult(k)
BigKci := Listen()	pi_inverse := Inverse(pi)
Verify(BigVi,BigKri)	BigKci.ScalarMul(pi_inverse)
BigKr=BigKri	Send((BigKri,BigVi,BigKci),Recipient)
}	End()

```
BigK:=BigKr.ScalarMul(kr)
End()
```

第一轮次如表 x-x 伪代码所示，签名者直接进入监听状态。接收者生成椭圆曲线随机数并初步计算随机点中间结果。接收者将随机点中间结果发送给其中一个签名者，然后进入监听状态。

收到消息的签名者首先从消息中提取随机点中间结果，然后选取一个随机数，将这个随机数与随机点中间结果进行倍点运算得到新的随机点中间结果，将这个新的结果发送给接收者后，进入下一轮次。

接收者收到新的数据后，重复进行第一轮次的操作，用自己的随机数再次进行倍点运算，然后发送给下一个签名者，等待下一个签名者的消息。

当接收者与所有签名者都交互了一次后，接收者进入下一轮次。

表 x-x 签名算法第二轮次伪代码

签名算法：Round2

接收者：	签名者：
mi := ConvertToShares(m)	while(Listen() == null){
r:=GetRFromBigK(BigK)	Wait()
for i:=0;i<Len(Signers);i++{	}
Cmi:=Encrypt(mi,key[i])	Data := Listen()
Cri:=Encrypt(r,key[i])	Cmi,Cri,Cmi_a,Cri_a,index
Beta := GetRandomInt()	Key:=GetByIndex(index)
Cmi_a:=HomoMul(Beta,Cmi)	Ci:=Calculate(Cmi,Cri,ki,Key)
Cri_a:=HomoMul(Beta,Cri)	Ci_a:=Calculate(Cmi_a,Cri_a,ki,Key)
Data :=(Cmi,Cri,Cmi_a,Cri_a,index[i])	Send((Ci,Ci_a),Recipient)
Send(Data,GetNextSigner())	
}	
while(IsRecipientFromAll()){	
Wait()	
}	
NextRound()	

第二轮次如表 x-x 伪代码所示，签名者直接进入监听状态。接收者按照算法对待签名消息进行加法分片，同态加密等操作构造出要发送给签名者的密文数据和验证数据。为每段数据确定好发送给签名者的顺序，然后将数据发送给每段数

据对应的第一位签名者，然后接收者进入监听状态等待签名者的消息。

当签名者接收到本轮此的消息后，按照算法对数据进行处理，并回发处理后的数据，然后进入下一轮次。

接收者接收到所有的签名者的消息后进入下一轮次。

第三轮次如表 x-x 伪代码所示，签名者直接进入监听状态。接收者从消息中提取出上一轮此签名者处理过的数据，按照算法对这些数据进行处理后将这些数据按照在第二轮次中确定的签名者顺序发给这段数据对应的下一个签名者，然后进入监听状态。

签名者接收到第三轮次的消息后，按照算法进行处理，并回发给接收者，接着进入监听状态。

接收者接收到签名者的消息后重复第三轮次开始时的操作，将处理过的消息发送给对应的下个签名者，直到所有消息都经过了所有签名者处理后进入下一轮次。签名者在这轮次也是重复监听，处理，发送的操作直到处理了所有签名消息后，签名者即完成了签名算法的所有步骤，结束算法。

表 x-x 签名算法第三轮次伪代码

签名算法：Round3	
接收者：	签名者：

Ci[],Ci_a[]:=Listen()
Repeat:
for i:=0;i<Len(Ci);i++{
　Verify(Ci[i],Ci_a[i])
　kr_inv:=inverse(kr)
　beta := GetRandomInt()
　Cix[i]:=HomoMul(kr_inv,Ci[i])
　Cix_a[i]:=HomoMul(beta,Cix[i])
　Send((Cix,Cix_a),GetNextSigner())
}
If(!IsAllOk())
　Goto: Repeat
End()
	Repeat
	while(Listen() == null){
	　Wait()
	}
	Cix,Cix_a := Listen()
	kp := ki_inv*pi
	Ci:=HomoMul(kp,Cix)
	Ci_a:=HomoMul(kp,Cix_a)
	Send((Ci,Ci_a),Recpient)
	If(!IsAllOk())
	　Goto: Repeat
	EndAlgorithm()

表 x-x 签名算法第四轮次伪代码

签名算法：Round4

```
接收者：
Verify(Ci[],Ci_a[])
Csi[]:=HomoMul(kr_inv*kr_inv,Ci[])
Smi[] :=Decrypt(Csi[])
Sm:=Sum(Smi)
p_inv := Inverse(prime_mask)
S:=p_inv*Sm
Signatrue:=(S,r)
if Verify(Signature) == true{
    Output Signature
    EndAlgorithm()
}else{
    IdentifyError()
}
```

第四轮次如表 x-x 伪代码所示，接收者对接收到的数据进行校验，解密，聚合，得到完整的签名。接着对签名进行验证，如果验证通过，结束签名算法。如果验证不通过，则需要运行错误识别程序，排查是哪个签名者导致了签名失败。

如表 4-7，根据这四个轮次的通讯的消息，我们可以定义六种消息类型。

表 x-x 签名者中的消息类型

消息名称	字段名称	类型	含义	对应符号
SignRound1Message1	BigKr_x	bytes	kr 对应的倍点 x 坐标值	K_r
	BigKr_y	bytes	kr 对应的倍点 y 坐标值	
SignRound1Message2	BigKri_x	bytes	随机点中间结果的 x 坐标	K_{r*}
	BigKri_y	bytes	随机点中间结果的 y 坐标	
	BigVi_x	bytes	验证点的 x 坐标	V_i
	BigVi_y	bytes	验证点的 y 坐标	
	BigKci_x	bytes	承诺点的 x 坐标	K_{Ci}
	BigKci_y	bytes	承诺点的 y 坐标	
SignRound2Message1	Cmi	bytes	消息分片 mi 的密文	c_{mi}
	Cri	bytes	r 的密文	c_{ri}
	Cmi_a	bytes	用于计算验证的密文	c'_{mi}

	Cri_a	bytes	用于计算验证的密文	c'_{ri}
	index	bytes	私钥索引	$index_i$
SignRound2Message2	Ci	bytes	签名中间结果的密文	c_i
	Ci_a	bytes	用于计算验证的密文	c'_i
SignRound3Message	Cix	bytes	签名中间结果的密文	c_{R^*}
	Cix_a	bytes	用于计算验证的密文	c'_{R^*}

第五章 系统测试

5.1 功能测试

我们使用 Go 语言自带的测试框架进行测试，使用不同的协程来运行不同的 LocalParty 来模拟不同参与者进行多方计算，协程之间通过通道进行信息交换。

因为三个算法都封装到了一个相同的系统框架中，所以我们对于三个算法的测试可以采用类似的方式，我们在 Go 语言的测试框架中编写了测试代码。

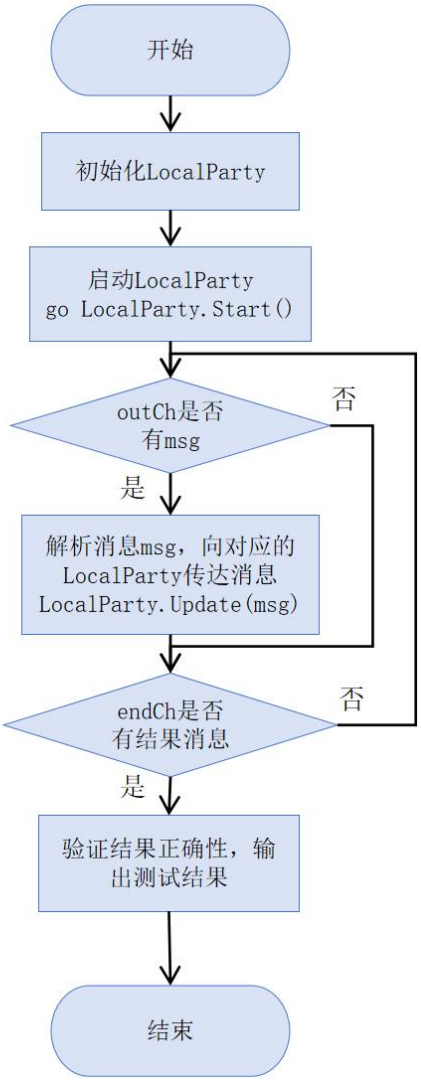


图 x-x 测试程序流程图、

如图 x-x，测试程序有以下步骤：

1、初始化四个 setup/keygen/signing 包的 LocalParty，其中三个为签名者，一个为辅助签名者/接收者。

2、使用 Go 协程调用所有参与者的 LocalParty.Start()函数启动算法。

3、进入监听循环，进行以下操作

4、持续监听四个 LocalParty 的 outCh 通道，当从通道中接收到消息时，解析消息内容确定消息的目的地，通过 LocalParty.Update(msg)向该 LocalParty 传入消息。

5、持续监听四个 LocalParty 的 endCh 通道，当监听到四个 LocalParty 都从 endCh 中传出算法结果时，跳出监听循环。

6、从算法结果中提取出通过共同计算的结果，针对不同的算法结果使用不同的方式检验结果的正确性，如果检验结果正确则测试通过。

如图 x-x 是签名者初始化算法的测试函数运行时打印的部分日志，对于该算法结果的检验方法为：

1、检查各个签名者是否生成了相同的掩盖因子，如果是，继续后续检验步骤，否则算法测试不通过。

2、取得各个签名者的掩盖因子分片。

3、对这些分片累乘，直接计算得到一个掩盖因子值为理论值。

4、将掩盖因子理论值与算法计算得到的掩盖因子对比，如果相同，则测试通过。

```
测试开始LocalParties启动
签名者1的掩盖因子分片pi
:14366994347996318236591856176785667012965609221523126577475051331513309521829276987030969543896856245449521887504878105486516286666857993532234126645322270002107386575889701393819106549366195476477031698292251490819145840052899211604020873502498025133286134945888428047210695447655167646744178859529254210524 7
计算的掩盖因子p:859425648781350065497192137370217288360722928038811360864753769324430972089 08

辅助签名者计算的掩盖因子p:85942564878135006549719213737021728836072292803881136086475376932443097208908

签名者3的掩盖因子分片pi
:14249172684514531505341557745766023151728370351942970583252317252500722197487898673942158498929115773191033089423100923854379792676644699238036847515176619937064457332413722851737752171875963562175356525679957821163452611852015179222969912562960736951614994750074845183467592710385300547738748063903682630829 9
计算的掩盖因子p:859425648781350065497192137370217288360722928038811360864753769324430972089 08

签名者4的掩盖因子分片pi
:16500859882337149074909929989418100275525392613902694711486742782250218086238360440755554687726125782382704744236381713138283291102603455942325611067251973586927600546079776512022541604139711450816959264459762976925322341009504697220381048092199467513808707477577118615312105055868077711435339571418 9199294867
计算的掩盖因子p:859425648781350065497192137370217288360722928038811360864753769324430972089 08

通过算法计算的掩盖因子:859425648781350065497192137370217288360722928038811360864753769324430972089 08
直接计算的掩盖因子     :859425648781350065497192137370217288360722928038811360864753769324430972089 08
两者相等，测试通过
```

图 x-x 初始化接收模块测试结果

如图 x-x 是接收者初始化算法的测试函数运行时打印的部分日志，对于该算法结果的检验方法为：

1、取得各个签名者的私钥分片，使用 Shamir 秘密共享方案提供的还原方法得到一个还原私钥。

2、将还原私钥与接收者秘密保存的私钥对比，如果相同，则测试通过。

```
=== RUN   TestLocalParty
    d:\FinalDesign\blind-tss-lib\ecdsa-blind\keygen\local_party_test.go:22: 测试开始
签名者1 私钥分片: 340078348329115689393311415034681425592186877541231498548955070467594836347 86
签名者2 私钥分片: 881598320743044182051150099319293297114668756226394564759708645780021141693 45
签名者3 私钥分片: 936443714666072128389568876685204201973259584692495022467384073203136798957 11
接收者私钥为: 915345740568868569763087660168908623464406389610605324687520414449715875103 97

使用私钥分片还原的私钥为:
915345740568868569763087660168908623464406389610605324687520414449715875103 97
----两者相等，测试通过-----
```

图 x-x 初始化算法模块测试结果

如图 x-x 是签名算法的测试函数运行时打印的部分日志，我们将算法结果中的签名，和接收者保存的公钥，以及待签名的消息输入到椭圆曲线签名验证算法中，观察是否通过签名验证。如果通过验证，则测试通过。

```
=== RUN   TestLocalparty

待签名消息m:556335219692082368716338813657858690783024564452331577534840252792865707433 19

启动LocalParty

签名完成，接收者算得签名:
r:694564746204206743653389676376068554690543650055822254956077114497500642399 43
s:797072415880943265303956773233578885821385973095647152394627503517001923863 48
接收者公钥为:
x坐标:805065657210376005158465385958760704657393970109913516372916664521085743 3420
y坐标:271410627019380922461811137864441754433025161670162729371222847660854413970 95
签名验证通过，测试通过
```

图 x-x 签名算法模块测试结果

5.2 性能测试

本节主要测试第四章中构建的系统性能，我们测试了 TB-ECDSA 方案中各个算法在不同阈值下的运行时间。随后，我们将 TB-ECDSA 方案与其他的几个门限签名方案进行了对比。

表 x-x 展示了 TB-ECDSA 各个算法在不同阈值下的运行耗时。随着参与者数量的增加，三个算法的运行耗时也随之增加。接收者初始化算法的主要运算在于 Shamir 秘密共享，其中的数学运算只涉及整数的乘法和加法，因此接收者初始化算法是三个算法中运行耗时最低的。另一方面，签名者初始化算法需要选取一个大素数，而大素数的生成算法很大程度上依赖于随机性，因此签名者初始化算法是三个算法中运行耗时最高的。

表 x-x　TB-ECDSA 方案各个算法运行耗时

签名者数量	签名者初始化(ms)	接收者初始化(ms)	签名(ms)
4	5772.542	0.689	148.575
8	17150.494	1.218	357.151
12	25080.287	1.557	623.568
16	34697.015	1.879	981.897
20	45148.160	2.702	1419.268

将表 x-x 中的数据绘制成折线图如图 x-x、图 x-x 和图 x-x 所示。

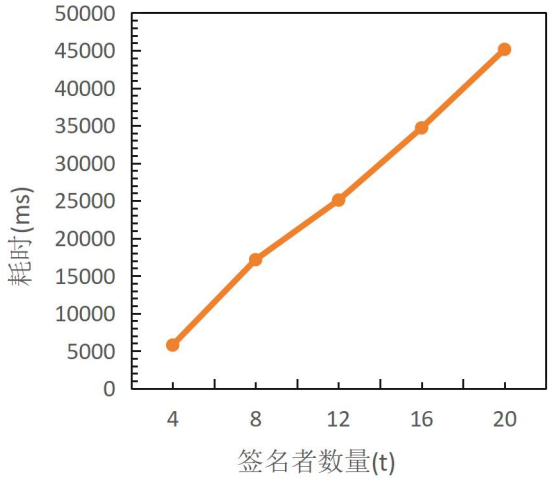


图 x-x　签名者初始化算法耗时与签名者数量关系图

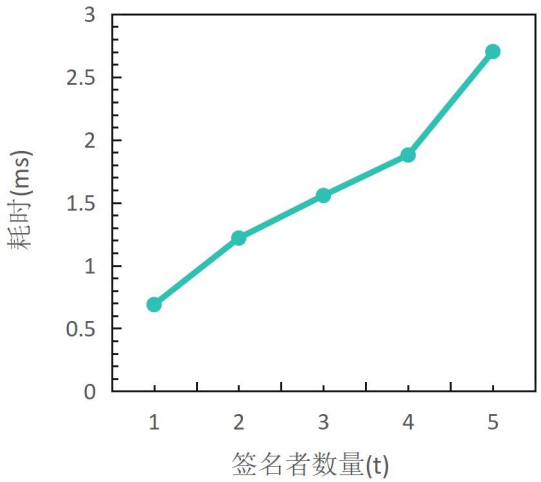


图 x-x　接收者初始化算法耗时与签名者数量关系图

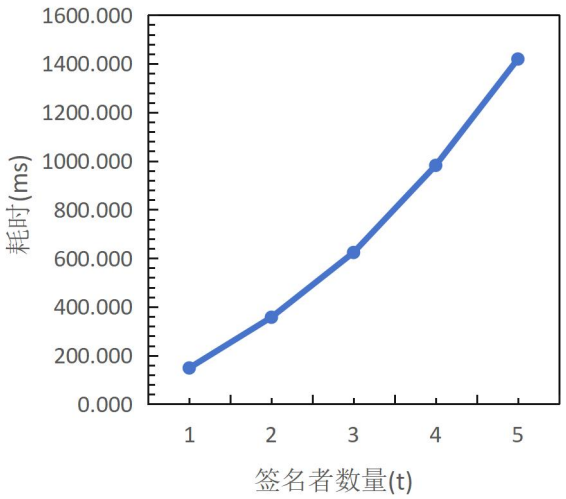


图 x-x 签名算法耗时与签名者数量关系图

从图 x-x 与图 x-x 中可以看出，签名者初始化算法与接收者初始化算法的算法耗时与签名者数量大致呈线性关系，而从图 x-x 中可以看出签名算法的算法耗时与签名者数量的曲线近似于二次函数，从而验证了第三章 3.8 节中对签名算法的计算复杂度分析。

我们将选用 GG18-ECDSA 方案、GG18-EdDSA 方案以及 FROST 方案与 TB-ECDSA 方案进行了算法的耗时比较。

表 x-x　门限签名方案比较

方案	盲性	用户匿名性	签名类型	错误识别	通信轮数
TB-ECDSA	是	是	ECDSA	是	密钥生成：1 签名：2t
GG18-ECDSA	否	否	ECDSA	是	密钥生成：4 签名：9
GG18-EdDSA	否	否	EdDSA	是	密钥生成：4 签名：9
FROST	否	否	Schnorr	是	密钥生成:3 签名：3

从表 x-x 中可以看出，在四种方案中，只有 TB-ECDSA 提供了盲性和用户匿名性；GG18-ECDSA 与 TB-ECDSA 都是产生 ECDSA 签名的方案；GG18-EdDSA 是 GG18 算法在 EdDSA[50]方案上的实现，理论上具有比 GG18-ECDSA 更高的运算效率；FROST 则是基于 Schnorr 签名，而 Schnorr 签名

具有线性特性，可实现更简单的多方签名流程。所有方案都支持错误识别。在通信轮次比较中，TB-ECDSA 是唯一动态轮次的方案，它的签名轮数与签名者数量 t 相关，而 FROST 由于 Schnorr 签名的线性特性，签名轮次最低。

表 x-x 各方案的接收者初始化/密钥生成耗时对比

签名者数量	TB-ECDSA	FROST	GG18-EdDSA	GG18-ECDSA
4	0.689	11.226	648.820	6551.029
8	1.219	19.732	3946.741	24229.541
12	1.558	28.216	11082.117	51682.493
16	1.879	65.152	25652.585	89775.416
20	2.703	132.645	52330.722	141302.878

在表 x-x 中我们对比了四个方案的接收者初始化算法或密钥生成算法的耗时，可以看出 TB-ECDSA 的初始化算法是同类算法中耗时最短的，其次是 FROST 方案，GG18-ECSDA 和 GG18-EdDSA 由于其在密钥生成阶段设计了复杂的范围证明，导致其密钥生成算法耗时都偏高。

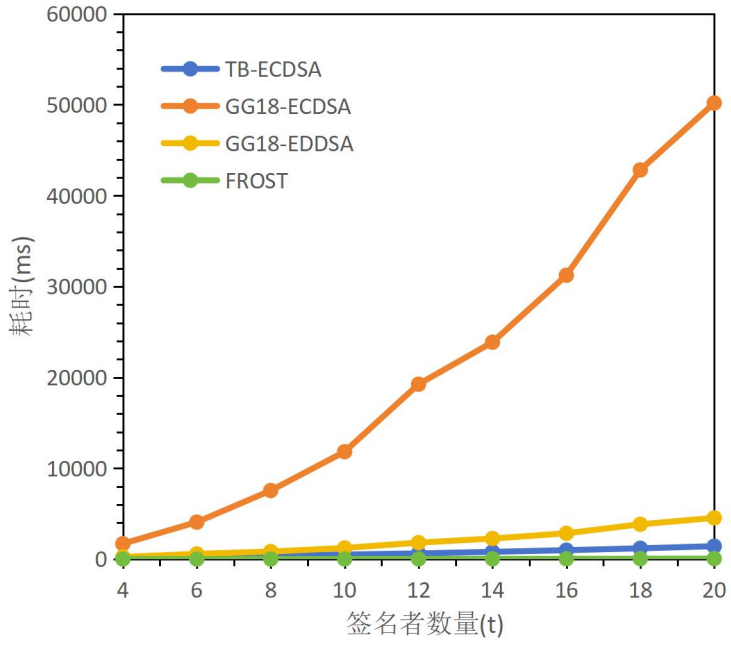


图 3-6 各方案签名算法耗时

图 3-6 为四种方案的签名算法在不同签名者数量下的运行耗时折线图，在我们的测试中，我们记录了四种门限签名方案在签名者数量从 4 逐步增加至 20 的

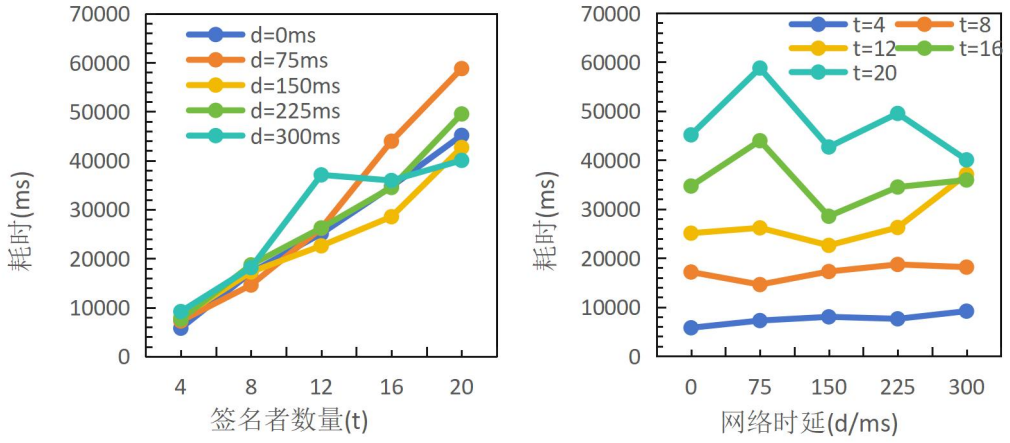
过程中的运算耗时。根据图 3-6 的显示结果，所有的方案的运算耗时都随签名者数量增加而增加。TB-ECDSA 的运算耗时始终低于 GG18-ECDSA 和 GG18-EdDSA。在签名者数量大于 4 之后，尽管 TB-ECDSA 的通信轮次比这两个方案更高，但当签名者数量的增加时，TB-ECDSA 运算耗时增加相比这两个算法更慢。FROST 方案由于采用 Schnorr 签名的性能优势以及更低的通信轮次，在四个方案中运行耗时最短。然而，与此相比，TB-ECDSA 在三个生成椭圆曲线签名的方案中的运行耗时表现与 FROST 最为接近。值得一提的是，TB-ECDSA 是四个方案中唯一提供了盲性和用户匿名性的。

5.3 网络拥塞测试

为了测试本系统在实际情况下的运行情况，我们通过给参与者引入网络时延，来测试方案中当某些参与者处于网络阻塞时对整个系统的耗时影响。我们首先分别对一位签名者和接收者处于网络拥塞的情况进行测试。我们对网络拥塞的参与者有如下定义，当一个参与者是处于网络拥塞的，那么他所发出的消息会经过时延 d 才可以到达接收方，同样其他参与者给该参与者发送的消息需要经过时延 d 才可以到达该参与者。我们设置了两种情景，一种是在算法中签名者处于网络拥塞中，另一种是接收者处于网络拥塞中。

5.3.1 签名者网络拥塞

1、签名者初始化算法：

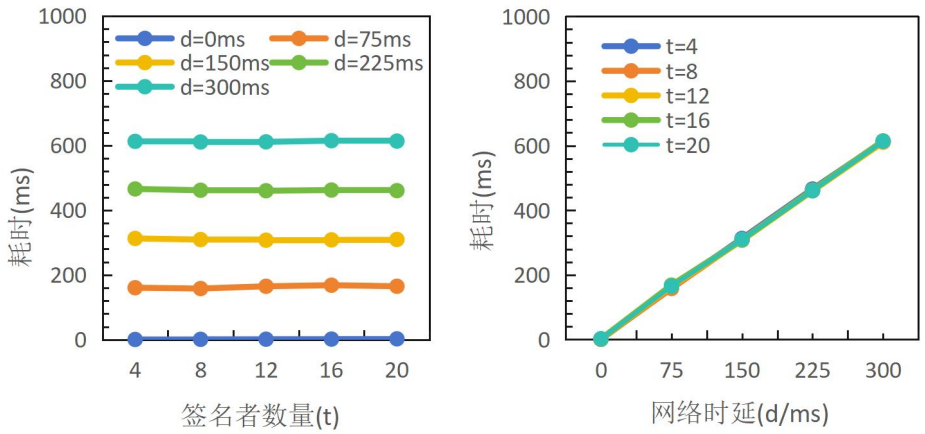


(a)耗时与网络时延关系图　　(b)耗时与签名者数量关系图

图 x-x 签名者网络阻塞时签名者初始化算法运行耗时

图x-x为有一名签名者在不同的网络时延下，签名者初始化算法的耗时情况。图 x-x(a)为不同网络时延下，算法耗时与签名者数量的关系图，图 x-x(b)为不同签名者数量下，算法耗时与网络时延的关系图。从图 x-x(a)可看出，随着签名者数量增加，总体的运行耗时大致都在增加。而不同网络时延对应的曲线在签名者数量较大时呈现了比较随机的大小关系。这一点在图 x-x(b)中有更明显地体现，在参与者数量为 16 和 20 时，随着网络时延的增加，耗时的增减具有随机性。这是因为在签名者初始化算法中，各个签名者需要生成一个大素数，而大素数的生成算法很大程度上依赖于随机性。当签名者数量增加，则进一步增大了耗时的随机性，导致算法耗时的分布范围远大于网络时延，这就导致了如图 x-x(B)中出现的即使网络时延较高但运行耗时反而更低的情况。因此可以得出结论网络时延在 300ms 以下时，签名者初始化算法的运行耗时更受算法本身以及签名者数量影响。

2、接收者初始化算法：

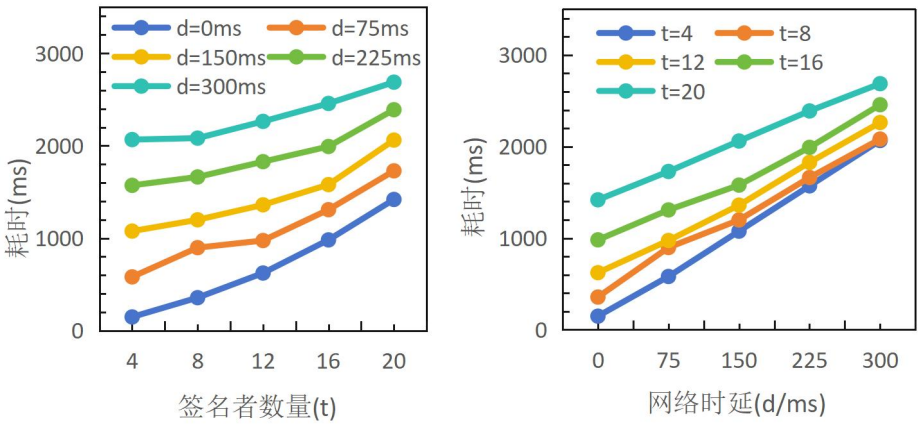


(a)耗时与签名者数量关系图　　(b)耗时与网络时延关系图

图 x-x　签名者网络阻塞时接收者初始化算法运行耗时

图x-x为有一名签名者在不同的网络时延下，接收者初始化算法的耗时情况。图 x-x(a)为不同网络时延下，算法耗时与签名者数量关系图，图 x-x(b)为不同签名者数量下，算法耗时与网络时延关系图。从图 x-x(a)可以看出，d=0ms 以外的曲线是 d=0ms 的曲线的平移。从图 x-x(b)中可以看出不同签名者数量的曲线几乎重叠在了一起，并且呈现明显的线性。因此可以得出结论，在接收者初始化算法中，签名者的网络时延会以常数叠加到算法耗时中。

3、签名算法：



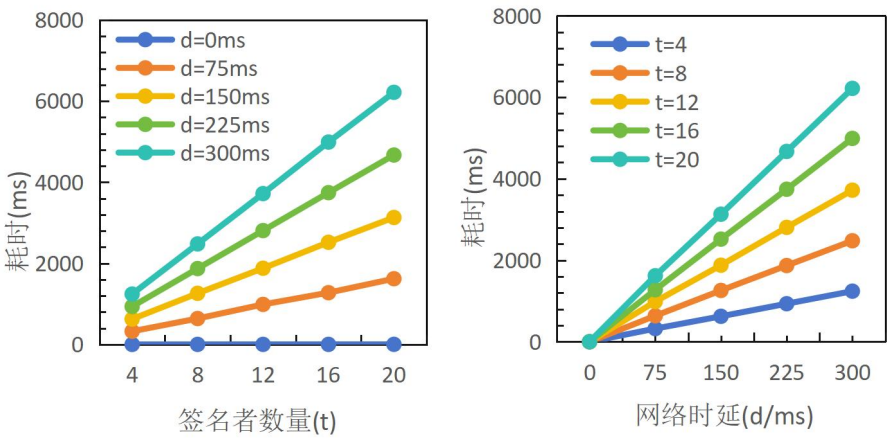
(a)耗时与签名者数量关系图　　(b)耗时与网络时延关系图

图 x-x 签名者网络阻塞时签名算法运行耗时

图 x-x 为有一名签名者在不同的网络时延下，签名算法的耗时情况。图 x-x(a)为不同网络时延下，算法耗时与签名者数量关系图，图 x-x(b)为不同签名者数量下，算法耗时与网络时延关系图。从图 x-x(a)中可以看出在不同网络时延下，算法耗时均随签名者数量而增加，而当网络延时较高时，算法耗时的增加幅度有减缓趋势。而从图 x-x(b)中可以看出，在不同的签名者数量下，算法耗时大致随网络时延线性增加，而当签名者数量较高，算法耗时的增加幅度有减缓趋势。综合图 x-x(a)与图 x-x(b)，可以而得出以下结论，算法耗时基本随着签名者的网络时延的增加线性增加，而当签名者数量比较大时，签名者的网络时延对算法耗时的影响相对较弱，算法本身的计算复杂度对算法耗时的影响相对较强。

5.3.2 接收者网络拥塞：

1、接收者初始化算法

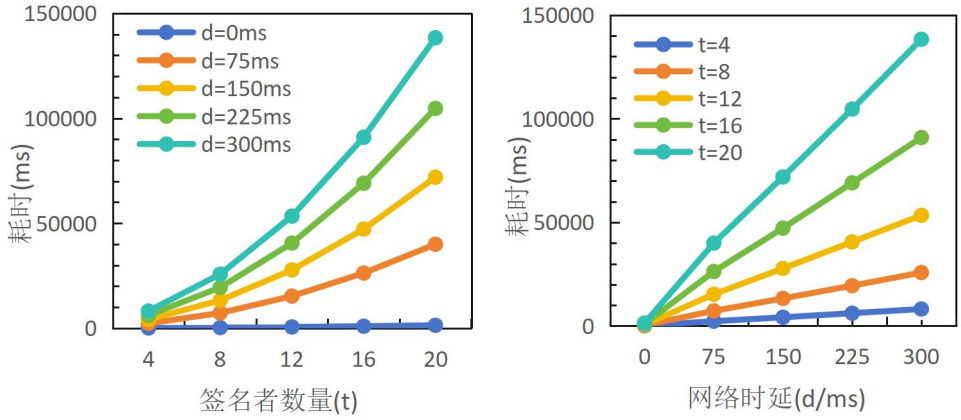


(a)耗时与签名者数量关系图　　(b)耗时与网络时延关系图

图 x-x 接收者网络拥塞时接收者初始化算法运行耗时

图x-x为有一名接收者在不同的网络时延下，接收者初始化算法的耗时情况。图 x-x(a)为不同网络时延下，算法耗时与签名者数量关系图，图 x-x(b)为不同签名者数量下，算法耗时与网络时延关系图。从图 x-x(a)中可以看出，在不同的网络时延下，算法耗时与签名者数量均呈线性关系，网络时延越大，算法耗时随签名者数量的增加幅度就越大。从图 x-x(b)可以看出，在不同的签名者数量下，算法耗时与网络延时呈线性关系，并且签名者数量越多，算法随网络延时的增加幅度就越大。因此可以得出结论，在接收者初始化算法中，当签名者数量越多时，接收者的网络时延会成比例地叠加到算法耗时中。

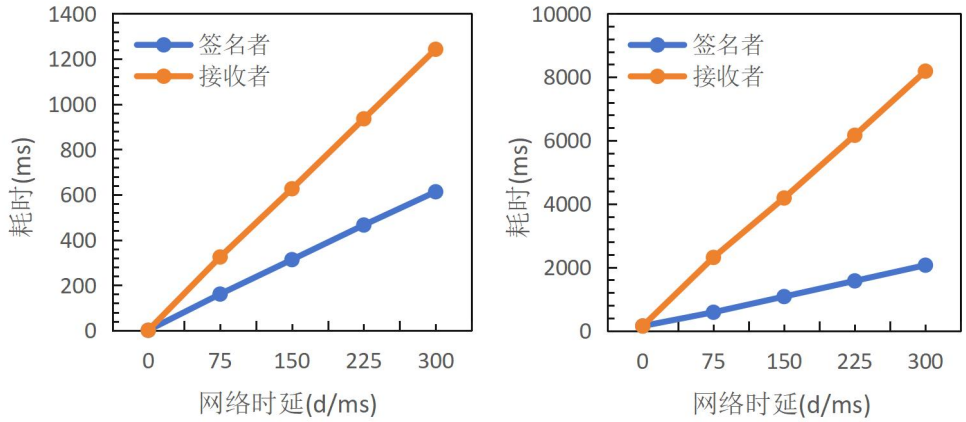
2、签名算法



(a)耗时与签名者数量关系图　　(b)耗时与网络时延关系图

图 x-x 接收者网络拥塞时签名算法的运行耗时

图 x-x 为有一名接收者在不同的网络时延下，签名算法的耗时情况。图 x-x(a)为不同网络时延下，算法耗时与签名者数量关系图，图 x-x(b)为不同签名者数量下，算法耗时与网络时延关系图。从图 x-x(a)中可以看出，算法耗时与签名者数量呈非线性关系，并且网络时延越高，算法耗时随签名者数量上升的幅度越大，结合第三章 3.8 节中对签名算法的通信复杂度分析，可以判断的此时算法耗时与签名者数量为二次函数关系，通信时延的增加相当于增大了二次项的系数。从图 x-x(b)中可以看出，算法耗时与网络时延成线性关系，并且签名者数量越多，算法随网络延时的增加幅度就越大。



(a)接收者初始化算法　　(b)签名算法

图 x-x　当 t=4 时算法耗时与网络时延的关系图

图 x-x 为签名者数量为 4 时，接收者初始化算法和签名算法的算法耗时与网络时延关系图，其中不同的曲线代表网络时延发生在不同的对象中。可以看出由于接收者在算法中负责中转签名者的消息，因此无论是接收者初始化算法，还是签名算法，接收者的网络时延对算法的耗时影响都是更加大的。

5.4 本章小结

在本章节中，我们对第四章中构建的系统进行了测试，主要分为功能测试、性能测试以及网络拥塞测试。

在功能测试中，我们通过一个统一的测试模板，针对不同算法设计了不同的测试结果验证方式，对系统中的三大模块进行了功能测试。

在性能测试中，我们首先测试了系统在不同签名者配置下的各算法运行耗时，并总结了各个算法的运行特点。随后我们引入了三个门限签名方案，将他们的密钥生成算法，和签名算法的耗时与系统的对应算法进行比较。TB-ECDSA 的接收者初始化算法是四个方案中最快的。而在签名算法测试结果显示，TB-ECDSA 相对于 GG18-ECDSA 和 GG18-EdDSA 在的运算效率更高。尽管 FROST 方案的运行时间最短，但在比较的几个门限椭圆曲线签名方案中，TB-ECDSA 的性能表现最接近 FROST，并且是唯一提供盲签名和用户匿名性的方案。

在网络拥塞测试中，我们分别设定了签名者和接收者在参与算法中的网络时延，并测试了在不同签名者数量下，不同算法的耗时表现。通过这些测试，我们

总结了签名者与接收者的网络时延对算法的运行耗时的影响特性。总的来说，由于接收者在方案中需要对签名者之间的消息进行中转，接收者的网络时延对算法的耗时影响更大。

第六章 总结与展望

6.1 总结

在当前的区块链生态系统中，数字签名技术扮演着关键角色，是其不可或缺的基础技术之一。个人或组织与区块链系统的互动都依赖于数字签名进行认证与验证。随着区块链应用的蓬勃发展，用户数量呈快速增长，但与之相伴而来的是现有私钥管理方案所带来的问题。私钥管理的疏忽可能导致丢失或泄露，使得区块链用户和组织失去对数字资产的控制权，造成重大损失。

为解决当前区块链用户私钥管理难题，MPC 钱包提供了一个解决方案，其核心技术是门限签名。然而，尽管门限签名技术已经部分解决了这一问题，但仍然存在着运算效率和隐私性方面的问题。因此，本文提出了一种名为 TB-ECDSA 的方案，该方案将门限签名与盲签名技术相结合，不仅提高了运算效率，还加强了用户的安全性和隐私性。本文的主要贡献包括：

1、提出了一项名为 TB-ECDSA 的创新签名方案，旨在解决当前 ECDSA 门限签名方案所存在的高运算成本和隐私泄露问题。TB-ECDSA 方案将门限签名和盲签名技术相结合，利用同态加密技术，有效地保护了用户的隐私，使签名者无法从签名过程中获取任何有用信息。并且，与主流的 ECDSA 门限签名方案相比，TB-ECDSA 方案在运算成本上具有明显优势。

2、在构建 TB-ECDSA 方案的过程中，本研究引入了用户匿名性的概念。该概念旨在确保多个节点在为多个用户提供服务时所产生的中间数据无法被关联。通过场景分析，我们总结实现用户匿名性的条件。本研究基于这些条件，构建了 TB-ECDSA 方案，从而在保障隐私的基础上提高了多方计算系统的安全性。

3、为了构建 TB-ECDSA 方案，本研究提出了 MSP 算法，解决多方各自持有秘密值且需计算秘密值乘积的问题。该算法通过将秘密值分片相乘并构造新的 Shamir 分片，使得节点能够在不泄露个体秘密值的情况下完成所有秘密值的乘积计算，从而为构建安全多方计算方案提供了新的数学工具。

4、基于 TB-ECDSA 方案，本研究实现了一个完整的门限盲签名系统。该系统基于开源密码库构建，包括签名者初始化模块、接收者初始化模块和签名模块。经过功能测试验证其正常运行后，本研究进行了性能测试，并将其与其他门限签

名方案进行了全面对比。实验结果表明，TB-ECDSA 方案具有较高的算法效率。此外，本研究还对系统的网络拥塞情况进行了评估，以研究网络拥塞对系统性能的影响，结果表明，由于接收者在方案中需要作为消息的中转，因此接收者端的网络拥塞对系统的整体影响更大。

6.2　展望

目前，对基于多方计算的 ECDSA 门限盲签名方案的研究相对较少，这可能是因为该领域的探索仍处于初级阶段。本文提出的 TB-ECDSA 方案填补了基于多方计算的ECDSA门限盲签名方案的研究空白。但受限于自己的专业知识局限，本文提出的方案依然有许多改进的空间和发展方向。

首先，目前在签名的过程中，签名者是明确知道自己是与哪些签名者进行交互生成签名的，这给共谋攻击提供了攻击方向。为此可以引入签名者之间的匿名性，使得每次签名的过程签名者都不知道自己与哪些签名者进行了交互，进一步提高系统的匿名性和安全性。

其次，当前的方案并没有设计动态添加或减少签名者集合的功能，这是出于安全性的考虑，因为增加或减少签名者集合都涉及了密钥分片的重新分配，需要设计协议保证这个过程不会泄露私钥的信息。倘若存在能够安全的完成添加或减少签名者集合的功能，结合签名者之间的匿名性就可以构建一个基于信用评价的去中心化签名者服务系统。接收者可以从集合中选择任意的签名者完成签名，诚实完成签名的签名者将获得正面评价，反之不诚实的签名者将会被踢出签名者社区。

最后，尽管本文提出的 TB-ECDSA 方案在门限盲签名系统的设计中迈出了重要的一步，但整体架构仍需进一步完善。目前的方案可能存在一些潜在的改进空间，包括但不限于提升系统的可扩展性、优化算法的效率、增强安全性等方面。未来的研究方向之一是如何进一步完善整个方案架构，以设计出更高效、更安全的方案。通过对现有方案的深入分析和改进，可以不断提升门限盲签名系统的性能和实用性，从而更好地满足实际应用的需求。

参 考 文 献

[1] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[2] Buterin V. Ethereum white paper[J]. GitHub repository, 2013, 1: 22-23.

[3] Monrat A A, Schelén O, Andersson K. A survey of blockchain from the perspectives of applications, challenges, and opportunities[J]. Ieee Access, 2019, 7: 117134-117151.

[4] Parisi C, Budorin D, Khalavka O. Wallet Security[M]//A Comprehensive Guide for Web3 Security: From Technology, Economic and Legal Aspects. Cham: Springer Nature Switzerland, 2023: 61-79.

[5] Aumasson J P, Hamelink A, Shlomovits O. A survey of ECDSA threshold signing[J]. Cryptology ePrint Archive, 2020.

[6] Shamir A. How to share a secret[J]. Communications of the ACM, 1979, 22(11): 612-613.

[7] Pedersen T P. A threshold cryptosystem without a trusted party[C]//Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques Brighton, UK, April 8–11, 1991 Proceedings 10. Springer Berlin Heidelberg, 1991: 522-526.

[8] Gennaro R, Jarecki S, Krawczyk H, et al. Secure distributed key generation for discrete-log based cryptosystems[C]//Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18. Springer Berlin Heidelberg, 1999: 295-310.

[9] Gennaro R, Jarecki S, Krawczyk H, et al. Secure applications of Pedersen's distributed key generation protocol[C]//Topics in Cryptology—CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003 Proceedings. Springer Berlin Heidelberg, 2003: 373-390.

[10] Desmedt Y. Society and group oriented cryptography: A new concept[C]//Conference on the Theory and Application of Cryptographic Techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987: 120-127.

[11] Shoup V. Practical threshold signatures[C]//Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19. Springer Berlin Heidelberg, 2000: 207-220.

[12] Lysyanskaya A, Peikert C. Adaptive security in the threshold setting: From cryptosystems to signature schemes[C]//Advances in Cryptology—ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9–13, 2001 Proceedings 7. Springer Berlin Heidelberg, 2001: 331-350.

[13] Damgård I, Koprowski M. Practical threshold RSA signatures without a trusted dealer[C]//International Conference on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001: 152-165.

[14] MacKenzie P, Reiter M K. Two-party generation of DSA signatures[J]. International Journal of Information Security, 2004, 2: 218-239.

[15] 王斌, 李建华. 无可信中心的 (t, n) 门限签名方案[J]. 计算机学报, 2003, 26(11): 1581-1584.

[16] 尚铭, 马原, 林璟锵, 等. SM2 椭圆曲线门限密码算法[J]. 密码学报, 2014, 1(2): 155-166.

[17] Gennaro R, Goldfeder S. Fast multiparty threshold ECDSA with fast trustless setup[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 1179-1194.

[18] Feldman P. A practical scheme for non-interactive verifiable secret sharing[C]//28th Annual Symposium on Foundations of Computer Science (sfcs 1987). IEEE, 1987: 427-438.

[19] Doerner J, Kondi Y, Lee E, et al. Secure two-party threshold ECDSA from ECDSA assumptions[C]//2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 980-997.

[20] Doerner J, Kondi Y, Lee E, et al. Threshold ECDSA from ECDSA assumptions: The multiparty case[C]//2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019: 1051-1066.

[21] Castagnos G, Catalano D, Laguillaumie F, et al. Bandwidth-efficient threshold EC-DSA[C]//IACR International Conference on Public-Key Cryptography. Cham: Springer International Publishing, 2020: 266-296.

[22] .Canetti R, Gennaro R, Goldfeder S, et al. UC non-interactive, proactive, threshold ECDSA with identifiable aborts[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. 2020: 1769-1787.

[23] Wong H W H, Ma J P K, Yin H H F, et al. Real Threshold ECDSA[C]//NDSS. 2023

[24] Schnorr C P. Efficient identification and signatures for smart cards[C]//Advances in Cryptology—CRYPTO'89 Proceedings 9. Springer New York, 1990: 239-252.

[25] Komlo C, Goldberg I. FROST: Flexible round-optimized Schnorr threshold signatures[C]//Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27. Springer International Publishing, 2021: 34-65.

[26] Chaum D. Blind signatures for untraceable payments[C]//Advances in Cryptology: Proceedings of Crypto 82. Boston, MA: Springer US, 1983: 199-203.

[27] Chaum D, Fiat A, Naor M. Untraceable electronic cash[C]//Advances in Cryptology—CRYPTO’88: Proceedings 8. Springer New York, 1990: 319-327.

[28] Pointcheval D, Stern J. Provably secure blind signature schemes[C]//International Conference on the Theory and Application of Cryptology and Information Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996: 252-265.

[29] Okamoto T. Provably secure and practical identification schemes and corresponding signature schemes[C]//Annual international cryptology conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992: 31-53.

[30] Stadler M, Piveteau J M, Camenisch J. Fair blind signatures[C]//Advances in Cryptology—EUROCRYPT’95: International Conference on the Theory and Application of Cryptographic Techniques Saint-Malo, France, May 21–25, 1995 Proceedings 14. Springer Berlin Heidelberg, 1995: 209-219.

[31] Abe M, Fujisaki E. How to date blind signatures[C]//International Conference on the Theory and Application of Cryptology and Information Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996: 244-251.

[32] Fuchsbauer G, Plouviez A, Seurin Y. Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model[C]//Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30. Springer International Publishing, 2020: 63-95.

[33] Schnorr C P. Security of blind discrete log signatures against interactive attacks[C]//International Conference on Information and Communications Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001: 1-12.

[34] Yi X, Lam K Y. A new blind ECDSA scheme for bitcoin transaction anonymity[C]//Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security. 2019: 613-620.

[35] Paillier P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999: 223-238.

[36] Qin X, Cai C, Yuen T H. One-more unforgeability of blind ecdsa[C]//Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part II 26. Springer International Publishing, 2021: 313-331.

[37] Hanzlik L. Non-interactive blind signatures for random messages[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham: Springer Nature Switzerland, 2023: 722-752.

[38] 陈倩倩, 秦宝东. 基于 SM9 的两方协同盲签名方案[J]. 计算机工程, 2023, 49(6): 144-153,161.

[39] Kuchta V, Manulis M. Rerandomizable threshold blind signatures[C]//Trusted Systems: 6th International Conference, INTRUST 2014, Beijing, China, December 16-17, 2014, Revised Selected Papers 6. Springer International Publishing, 2015: 70-89.

[40] Crites E, Komlo C, Maller M, et al. Snowblind: A threshold blind signature in pairing-free groups[C]//Annual International Cryptology Conference. Cham: Springer Nature Switzerland, 2023: 710-742.

[41] Juang W S, Liaw H T. Fair blind threshold signatures in wallet with observers[J]. Journal of Systems and Software, 2004, 72(1): 25-31.

[42] Juang W S, Lei C L. Partially blind threshold signatures based on discrete logarithm[J]. Computer Communications, 1999, 22(1): 73-86.

[43] 陆洪文, 郑卓. 基于双线性对的门限部分盲签名方案[J]. 计算机应用, 2005, 25(9): 2057-2059.

[44] Makriyannis N, Yomtov O, Galansky A. Practical key-extraction attacks in leading mpc wallets[J]. Cryptology ePrint Archive, 2023.

[45] Aki. Digital signatures: a tutorial survey[J]. Computer, 1983, 16(2): 15-24.

[46] Johnson D, Menezes A, Vanstone S. The elliptic curve digital signature algorithm (ECDSA)[J]. International journal of information security, 2001, 1: 36-63.

[47] Paillier P. Public-key cryptosystems based on composite degree residuosity classes[C]//International conference on the theory and applications of cryptographic techniques. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999: 223-238.

[48] Boneh, Dan. "The decision diffie-hellman problem." International algorithmic number theory symposium. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[49] Barthe G, Daubignard M, Kapron B, et al. Computational indistinguishability logic[C]//Proceedings of the 17th ACM conference on Computer and Communications Security. 2010: 375-386.

[50] Bernstein D J, Duif N, Lange T, et al. High-speed high-security signatures[J]. Journal of cryptographic engineering, 2012, 2(2): 77-89.

[51] Anderson R, Needham R. Robustness principles for public key protocols[C]//Annual International Cryptology Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995: 236-247.

[52] Bellare M, Sandhu R. The security of practical two-party RSA signature schemes[J]. Cryptology ePrint Archive, 2001.

[53] Lindell Y. Fast secure two-party ECDSA signing[C]//Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part II 37. Springer International Publishing, 2017: 613-644.

[54] Micali S, Goldreich O, Wigderson A. How to play any mental game[C]//Proceedings of the Nineteenth ACM Symp. on Theory of Computing, STOC. New York, NY, USA: ACM, 1987: 218-229.

[55] Wagner D. A generalized birthday problem[C]//Annual International Cryptology Conference. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002: 288-304.

[56] Kim M, Cho S, Choi S, et al. A key recovery protocol for multiparty threshold ecdsa schemes[J]. IEEE Access, 2022, 10: 133206-133218.

[57] ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms[J]. IEEE transactions on information theory, 1985, 31(4): 469-472.

[58] Kaur R, Kaur A. Digital signature[C]//2012 International Conference on Computing Sciences. IEEE, 2012: 295-301.

[59] Gadekallu T R, Huynh-The T, Wang W, et al. Blockchain for the metaverse: A review[J]. arXiv preprint arXiv:2203.09738, 2022.

[60] Cao J, Weng J, Pan Y, et al. Generalized attack on ECDSA: known bits in arbitrary positions[J]. Designs, Codes and Cryptography, 2023, 91(11): 3803-3823.

[61] Wang Z, Chaliasos S, Qin K, et al. On how zero-knowledge proof blockchain mixers improve, and worsen user privacy[C]//Proceedings of the ACM Web Conference 2023. 2023: 2022-2032.

[62] Guo H, Yu X. A survey on blockchain technology and its security[J]. Blockchain: research and applications, 2022, 3(2): 100067.

附录 A 签名算法阶段二执行示意图

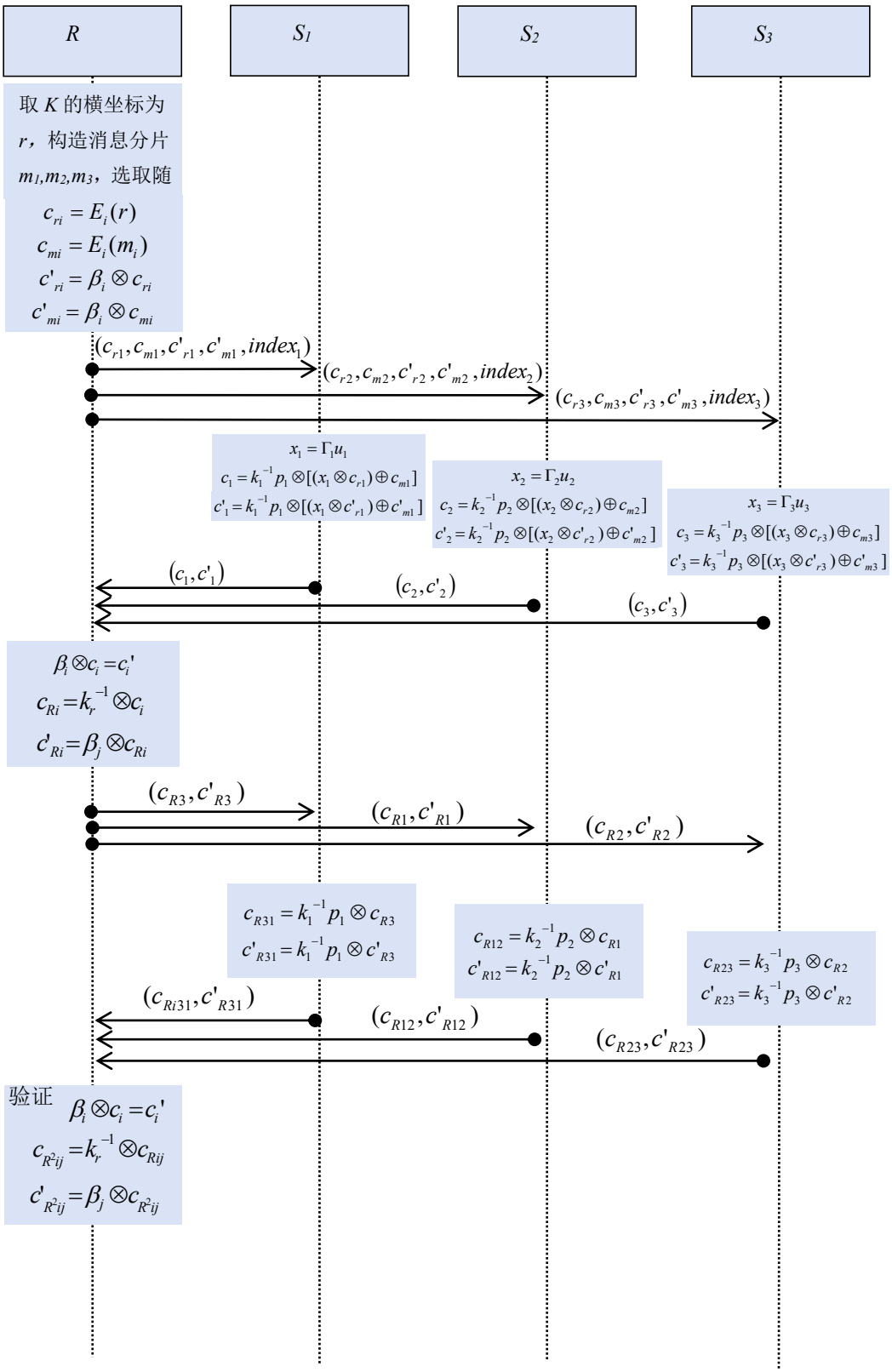


图 A-1 签名算法阶段二步骤 1~14 执行示意图

附录 B 签名算法阶段二执行示意图

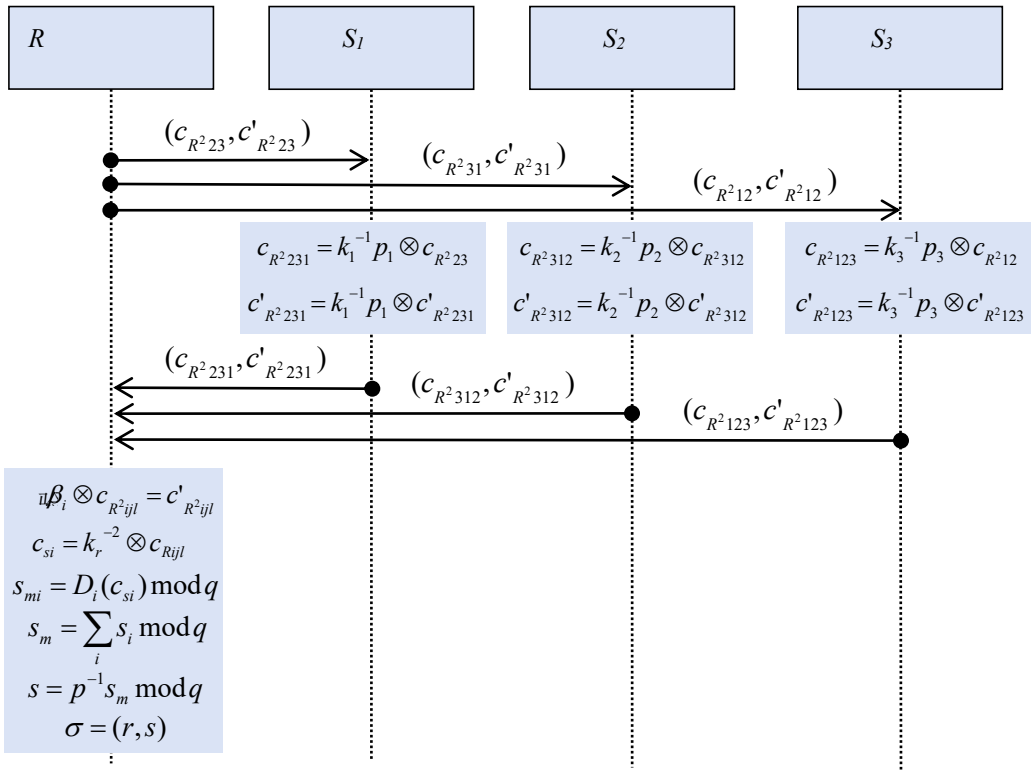


图 B-1　签名算法阶段二重复步骤 8~14 一次以及步骤 16~19

致　谢

三年的研究生生活转瞬即逝，回头看看，加上本科我已经在深大八年了。这八年我见证了荔园的老建筑倒下和新建筑升起，而荔园也见证了我从当初第一次踏入校园的青涩模样成长到如今的样子。我感谢深大这位老朋友，感谢它曾带给我的欢乐和难过，感谢它曾留给我的遗憾和释怀，感谢它为我求学的道路上点亮了一路的明灯。

在三年的研究生生涯中，我深深感激导师王滔滔老师给予我的悉心指导与帮助。滔滔老师时刻强调着及时沟通的重要性，在我遇到技术难题时，他总是毫不吝啬地给予帮助与耐心指导。实验室的环境也是一直以来的支撑所在，深圳大学区块链研究中心的老师们不仅知识渊博，而且开明豁达，他们的悉心指导使我受益匪浅。同学们更是团结友爱、努力上进，我们相互鼓励、相互成长。每次汇报，张胜利老师、王滔滔老师、杨晴老师给予我的精准意见都是宝贵的财富。他们的建议不仅帮助我解决了困惑，更让我感受到了自己的成长与进步。在王老师的悉心指导下，我度过了充实而难忘的研究生生涯，我将倍加珍惜这段宝贵的经历，并努力将所学知识付诸实践。

感谢我的父母和家人，我之所以能求学至今是他们在背后默默地付出和支持。感谢他们一以贯之地尊重和支持我的选择，让我能够去做自己想做的事情。也感谢他们为我提供的经济条件，让我无需考虑生活的压力地投入到校园的学习生活中。

在过去的三年研究生生活中，我最多的陪伴来自于实验室的伙伴们以及舍友们。我要特别感谢那两位与我共同度过朝夕相处的舍友兼实验室伙伴，朱俊杰和王紫阳。同时也要感谢与我同在一个课题组的伙伴们，包括梁立铭、李炫晞、夏嵩、杨明钦、尤肖男、石佳怡、刘锦谦、马千里、郭威、王真、宁新威、以及方波。他们的陪伴与支持让我的研究生生活充满了温暖和动力。同时，我也要特别感谢林子彬师弟对我在研究课题上的帮助与支持。

我要由衷地感谢那些在繁忙之余抽出宝贵时间对我的论文进行评审的专家老师们。您们的辛勤努力与专业意见是我前行路上最宝贵的支撑与鼓励。