# 摘要

略略略

**关键词：**门限签名，盲签名，ECDSA，同态加密

# ABSTRACT

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.……

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX.……

……

**Key words:** Threshold Signature, Blind Signature, ECDSA, Homomorphic Encryption

# 目录

目录

[摘要 1](#_Toc3488)

[ABSTRACT 2](#_Toc29078)

[目录 3](#_Toc12460)

[第一章 绪论 4](#_Toc1395)

[1.1 论文研究背景及意义 4](#_Toc6200)

[第二章 相关技术和理论介绍 4](#_Toc5905)

[2.1 Shamir秘密共享 4](#_Toc30139)

[2.2 ECDSA数字签名方案 5](#_Toc1545)

[2.3 Paillier同态加密修改方案 5](#_Toc22125)

[第三章 基于ECDSA的门限盲签名协议 7](#_Toc3786)

[3.1符号系统 7](#_Toc8390)

[3.2 基于ECDSA的门限盲签名协议 7](#_Toc21132)

[3.3 正确性分析 13](#_Toc31545)

[3.4 安全性分析 13](#_Toc10152)

[3.5 复杂度分析 13](#_Toc19875)

[第四章 基于ECDSA的门限盲签名系统实现 13](#_Toc15819)

[第五章 总结与讨论 13](#_Toc23355)

# 第一章 绪论

## 论文研究背景及意义

2008年，中本聪提出了比特币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，则见证了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。随之而来的是越来越多的用户开始涉足区块链世界。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由各个区块链节点进行共识以完成一笔转账操作，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。

为了使用户更便利地管理其数字资产，数字钱包等产品应运而生，成为用户与区块链之间的桥梁。然而，随着区块链技术的普及和用户群体的扩大，安全隐患也逐渐暴露出来。

遗失私钥或私钥被黑客攻击是区块链用户面临的重大风险之一。许多用户可能因个人疏忽，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。更为严重的是，黑客可能利用各种技术手段入侵数字钱包或交易所，窃取用户的私钥，进而窃取用户的数字资产，给用户带来巨大的经济损失。//数据或新闻引用

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。因此，区块链技术领域亟需加强安全意识和技术防范手段的研发，以应对不断演变的安全威胁，为用户提供更加可靠的数字资产保障。

# 相关技术和理论介绍

## 2.1 Shamir秘密共享

Shamir秘密分享是由Adi Shamir在[]中提出的一个算法。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需要的最小部分数量。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，以为常数项构造一个次多项式:

(2-1)

则对于不同的取值，如可以计算出不同的函数值，这些函数值与其自变量共同构成一个秘密分片值，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:

(2-2)

(2-3)

## 2.2 ECDSA数字签名方案

数字签名算法DSA是1991年由Karivtz在文献[]中提出，随后在1994年由NIST修改成数字签名标准。而ECDSA是数字签名算法的一种变体，在区块链和加密货币中受到广泛应用。一个椭圆曲线数字签名方案包括四个部分。

1. 公共参数。
   1. 一个阶为的椭圆曲线循环群，以及其生成元，其中为素数。
   2. 哈希函数，可以将任意类型的消息M映射到
2. 密钥生成算法：输入一个安全参数，输出一对公私钥。私钥为从中随机选取的整数，公钥为。
3. 签名生成算法：输入一个任意的消息,输出签名。
   1. 计算。
   2. 随机选择整数。
   3. 计算，取的横坐标为。
   4. 计算。
   5. 输出。
4. 签名验证算法：输入，和。
   1. 验证和是否都属于。
   2. 计算。
   3. 取的横坐标为，如果则签名验证通过，否则该签名无效。

## 2.3 Paillier同态加密修改方案

### 2.2.1 Paillier同态加密方案

Paillier同态加密方案[]是一个基于离散对数困难问题构建的加密方案，它具有加法同态的性质，他的密钥对生成过程如下：

1. 随机选取两个大素数，需要满足条件与互素，不满足则重新选择随机数。
2. 计算，其中表示最小公倍数。
3. 随机选取一个正整数，要求并存在，若不存在，则需要重新选取,其中：

(2-4)

其中的除法表示向下取整。

得到的公钥为，私钥为，对于要加密的消息，它需要满足，它的加密过程如下：

1. 随机选取一个正整数，要求，且还需满足与互素。
2. 计算密文如下：

(2-5)

对于密文，解密过程为：

(2-6)

使用表示对进行加密得到的密文，对于任意的消息，Paillier加密方案具有以下性质：

(2-)

(2-)

使用表示密文的模相乘，使用表示密文的模指数运算，则上述性质可以写作：

(2-)

(2-)

则可以将定义为Paillier的同态加运算，定义为Paillier的同态数乘运算。上述性质即为Paillier加密方案的同态加法性质即密文的相加等于明文相加后再加密。

### 2.2.2 Paillier同态加密修改方案

Paillier同态加密因其优秀的性质而得到广泛应用，然而在文献[]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[]在此基础上提出了一个Paillier同态加密的修改方案，它的密钥生成如下：

1. 随机选取两个大素数，需要满足条件与互素，不满足则重新选择随机数。
2. 计算，其中为ECDSA中的公共参数椭圆曲线群的阶。

得到的公钥为，私钥为，对于要加密的消息，它需要满足，它的加密过程如下：

1. 随机选取一个正整数，要求，且还需满足与互素。
2. 计算密文如下：

(2-)

对于密文，解密过程为：

(2-)

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。

# 基于ECDSA的门限盲签名协议

## 3.1符号系统

## 3.2 基于ECDSA的门限盲签名协议

### 3.2.1 引言

假设接收者是诚实的，可信的，把接收者看作是用户。

本文提出的基于ECDSA的门限盲签名协议具有以下几个特点：

1. 门限签名：系统中允许有位签名者，当位签名者同意参与签名时才可以产生合法的签名，其中,满足。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。
2. 盲签名：参与签名的签名者无法获知签名的具体消息内容。通过使用同态加密技术，签名者在签名过程中只能对经过同态加密的密文进行同态运算处理，无法还原出原始消息的内容。这种消息的盲性保护了签名的机密性和隐私性，确保消息内容不会被签名者获取。
3. 私钥分片间的不可链接性：防止签名者推断出私钥分片之间的对应关系。假设签名者持有多个从属不同私钥的私钥分片，在本协议中签名者无法确定相互之间的私钥分片的对应关系。这将增大签名者进行共谋窃取私钥的难度，提高了系统的安全性。
4. 签名的不可链接性：防止签名者推断出实际签名过程与后来公开的签名之间的确切对应关系。不可链接性对于维护请求者的隐私和匿名性非常重要。通过防止签名者将签名过程与最终签名联系起来，请求者的身份和签名消息的内容保持机密。这个特性确保签名者无法追溯请求到具体签名，从而保护请求者免受潜在的隐私泄露或重新识别的风险。
5. 错误识别：如果签名者在签名协议中偏离协议，不按照协议规定的方式计算签名而导致最终产出的签名不合法，接收者可以通过签名产生的中间变量结合公开的参数来定位是哪个签名者的过失导致了签名不合法。

### 3.2.2 签名者初始化算法

假设所有位签名者中的位组成签名者集合，签名者表示为，每个都要在初始化的过程中得到一个属于自己的保密掩盖因子分片，以及一个公共参数掩盖因子，其中：



进行一次初始化还需要一个协调签名者进行，协调签名者可以由集合以外的任意成员担任。一次初始化的过程如下：

公共参数：

1、每个或都有一个正整数或作为自己的唯一标识。

步骤：

1. 随机选取一个大素数，作为掩盖因子分片。
2. 以为常数构造一个一次多项式，其中是随机选取的系数。
3. 计算，暂存。
4. 计算和，将发送给，将发送给
5. 或收到其他签名者发送来的或，计算



或



1. 或计算或。其中:

(3-)

1. 或向参与初始化的签名者公布或。
2. 或收到和后计算。
3. 计算，公开和掩盖因子，完成初始化。

一次初始化流程所产生的和与当前参与流程的签名者集合对应，因此对于个签名者，阈值为的系统，需要进行次初始化过程。各个签名者持有个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。

我们对这个初始化流程做一些原理解释。在步骤5中计算的(类似)可以展开成如下形式：



其中是由，这两个常数的组合相乘构成的常数，我们并不关心的具体值。式()的形式可以看作是一个多项式函数在自变量为时的取值，并且这个多项式的常数项为，即如下式：





因此可以将或看作是关于的一个Shamir秘密共享分片，它的阈值为。然后在步骤6中，利用将或转换乘了加法分片或。在所有签名者在步骤7公布自己的加法分片，而或中是不会泄露各个签名者的的。最后各个签名者就可以通过对这些加法分片求和来得到即掩盖因子。

### 3.2.3 接收者初始化

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

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。

接受者拥有私钥和对应公钥*，*接收者密钥分发步骤如下：

1. 以为秘密构造一个Shamir秘密共享，阈值为t，秘密分片为。
2. 记录，分别将发给。
3. 接收到后，存储并发送向发送存储索引。
4. 保存索引，删除，将秘密备份完成注册。

### 3.2.4 签名协议

#### 3.2.4.1 阶段一

签名协议的阶段一目的是为了生成椭圆曲线签名中的随机点，在这个过程中只有接收者最终得知了，签名者参与的生成但是无法得知的真实值。签名协议阶段一步骤如下：

公共参数：

1、椭圆曲线群的相关参数

私有参数：

1. 签名者:
   1. 掩盖因子分片

协议步骤：

1. 选择一个随机数，随机数。
2. 计算。
3. 计算。
4. 发送给。
5. 接收到，后，随机选择一个，计算：







发送给。

1. 接收到后，计算，如果成立，继续协议，否则中止协议。
2. 计算，将作为步骤2中的，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。
3. 完成步骤7会得到和
4. 计算。

我们对阶段一做一些详细的解释。步骤1~4中，接收者选择了一个随机数，然后接收者对随机数做倍点运算得到签名随机点的中间结果，同时接收者再用另外一个随机数与倍点运算得到，然后将和发送给一个签名者。

步骤5中，签名者接收到相关数据后将他选的随机数与及做相同的倍点运算得到和，再将随机数与掩盖因子分片的逆的乘积做倍点运算得到，再将发给接收者。

步骤6中，接收者收到数据后，计算，验证。如果验证不通过说明签名者偏离了协议，没有依照协议对做指定的运算，此时中止当前协议。如果检查通过了则继续进行协议，接收者保存，以便进行后续的验证。

步骤7中，签名者在将中间结果发送给下一个签名者继续处理前，需要用自己的随机数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。接收者对中间结果完成处理后，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中间结果进行了倍点运算。此时接收者得到的依旧是一个中间结果，需要对这个结果使用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。经过上述协议后得到的的具体值为：

(3-)

那么对应的为：



虽然在步骤4中首先将数据发送给了处理，实际上接收者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。

#### 3.2.4.2 阶段二

签名协议阶段二的目的是继续完成签名s。在阶段二中，接收者会与签名者们共同完成的计算，接收者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，再最终组合成。

公共参数：

1. 掩盖因子

私有参数：

1. 接收者
   1. 个Paillier加密方案的私钥
   2. 个Paillier加密方案的公钥
   3. 私钥分片的倍点
   4. 私钥分片对应的索引
   5. 随机点
   6. 待签名的消息
2. 签名者
   1. 私钥分片
   2. 掩盖因子分片
   3. 随机数

协议步骤：

1. 接收者取的横坐标为。
2. 接收者对要加密的消息进行加法分片得到份，其中满足



1. 接收者使用个Paillier加密方案加密和得和。
2. 接收者选取个随机数，计算和。
3. 接收者选取个随机数，计算。
4. 接收者将发送给。
5. 签名者接收到后，根据找到对应的，然后计算签名分片中间结果：



(3-)



1. 签名者将发送给接收者。
2. 接收者接受到，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。

1. 接收者计算



1. 接收者重新选取个随机数，计算



1. 接收者重新选取个随机数，计算。
2. 接收者将发送给。
3. 签名者收到后，计算：





1. 签名者将发送给接收者。
2. 接收者接收到。
3. 将作为步骤8中的，重复步骤9~15，直到所有签名分片中间结果被所有签名者都处理了一遍，在步骤13中，每次都需要发送给一个未处理过该部分签名分片的签名者。
4. 完成步骤17接收者会得到份经所有签名者处理过的中间结果
5. 接收者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。

1. 接收者计算：









21、接收者组合出完整的签名

我们对阶段二的步骤做一些详细的解释。在阶段二中，首先步骤1~2由接收者完成的计算以及对消息的分片，分片的结果对应每个签名者。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。

接着在步骤3中，接收者使用不同的Paillier公钥加密和得到。在这一步中使用了不同的Paillier公钥加密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。同时由于后续需要一起进行同态运算，所以消息分片需要与在同一个公钥下加密。

在步骤4中，接收者选取了个随机数与进行同态数乘运算得到。的作用是使得后续步骤9中接收者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5中，接着接收者再选取个随机数与Paillier公钥相乘得到。的作用是给签名者提供同态运算的公钥，如果直接提供公钥，后续相关的加密数据发送到下一个签名者时，他就收到了一部分与上一个签名者相同的数据，违反了原则\_因此要使用对真正的公钥进行掩盖。

在步骤7中，签名者接收到了加密的和即，以及其他用于校验和辅助同态运算的数据。签名者首先将它的密钥片段转换成加法分片，然后进行了进行了式(3-)的运算得到了，此处可以将写成，即是明文的密文。其中括号外的因子中是当前签名者在阶段一中选择的随机数，是在初始化阶段确定的掩盖因子分片。因子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。与经过了相同的运算，其对应的明文是。它只与相差了一个因子。

步骤9中接收者对和进行验证计算，验证是否由同态数乘所得。如果验证不通过说明签名者没有依照协议对和做指定的运算，此时中止当前协议。如果检查通过了则继续进行协议。

步骤10，接收者使用与作同态乘法得到。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，此时的明文为；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，保证原则\_得到满足。

步骤11~12的作用与步骤4~5类似。

步骤13，将发送给下一个签名者，这个签名者需要满足他从未处理过当前签名的分片数据。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，它对应的明文是，从左边的因子可以看出和都以及处理过，因为他们都把自己的秘密随机值乘入这个数据了，因此这个不可以再发给或进行处理，只能发给除他们以外的任意一个签名者。接收者可以根据每次数据交互的对象来记录每个分片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14~步骤15，签名者使用同态运算将乘入和后回发给接收者。在这一步中，乘入是为了完成签名，而的作用是为了掩盖。的作用具体而言可以从接收者视角进行分析，在步骤13中，接收者发送给签名者，在步骤16中，接收者接收到了来自处理过的。此时接收者可以对和进行解密获得明文，假设的明文为,那么的明文为,接收者对两个明文进行以下运算：



而正是乘入的数值，假如没有这个因子，那么接收者可以直接分析和得知的秘密值，这样是不安全的。因此需要引入一个掩盖因子，在接收者未知和的情况下，从计算出是相对困难的。

步骤18~19的作用类似步骤8~9。

步骤20，接收者已经获得了所有签名分片数据，此时这些签名分片数据的明文是



与真正的签名分片还差并且还需要去除掩盖因子。因此式()的作用是将运算到签名中，因此有：



此时得到的是依旧被掩盖因子遮盖的签名分片。将分片求和得到被掩盖的签名：



式()是使用公开的掩盖因子去除掩盖因子得到，即下式



再结合式()则可将化为：



与ECDSA数字签名方案中的式()一致，说明最终结果是一个有效的ECDSA数字签名。

### 3.2.5 签名验证和错误识别

#### 3.2.5.1签名验证

由上述签名算法生成的签名与ECDSA数字签名算法的签名形式是一致的，因此验签方式与ECDSA一致，接收者只需要计算



取的横坐标位，验证是否成立。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。

#### 3.2.5.2错误识别

当接收者发现生成的签名无法正常验签时，可以通过签名协议中的中间数据来排查出偏离协议导致签名失效的签名者。本协议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：如果签名者在签名协议阶段二步骤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签名。
  2. 没有使用正确的掩盖因子分片。
  3. 没有使用阶段一所选取的随机数。

则接收者可以通过以下步骤排查出该签名者：

1. 计算。
2. 验证下式是否成立。

()

假如成立，说明在步骤7没有出现偏离协议，否则说明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，应该有如下取值：



我们对式()等号两边分别逐级展开如式()和式()。





可以看出当签名者遵循协议流程时，式()左右两边应该是相等的。而当签名者偏离了协议时会导致，使得式()不成立。

2、第二类偏移协议：如果签名者在签名协议阶段二步骤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。
  2. 没有使用阶段一所选取的随机数。

则接收者可以通过以下步骤排查出该签名者；

1. 计算，。
2. 验证下式是否成立。



假如成立，说明在步骤14中没有出现偏离协议，否则说明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，和应该有如下取值：





我们对式()等号两边分别逐级展开：





可以看出当签名者遵循协议流程时，式()左右两边应该是相等的。而当签名者偏离了协议时会导致，使得式()不成立。

## 3.4 安全性分析

## 3.5 复杂度分析

# 基于ECDSA的门限盲签名系统实现

# 总结与讨论