密码学库功能说明书

V1.0

负责人：丁振洋

小组成员：朱焱、冯仲达、杨洋、丁振洋、张天逸

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 提交日期 | 主要编制人 | 审核人 | 版本说明 |
| 1.0 | 2022-10-30 | 密码学小组全体成员 | 丁振洋 | 文档初步编写 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1 引言 1](#_Toc122895073)

[1.1 编写目的 1](#_Toc122895074)

[1.2 项目背景 1](#_Toc122895075)

[1.3 相关术语 1](#_Toc122895076)

[1.3.1 哈希函数 1](#_Toc122895077)

[1.3.2 数字签名 1](#_Toc122895078)

[1.3.3 密钥交换 1](#_Toc122895079)

[1.3.4 秘密承诺 1](#_Toc122895080)

[1.3.5 秘密分享 1](#_Toc122895081)

[1.3.6 可验证随机数 1](#_Toc122895082)

[1.3.7 验证延迟函数 1](#_Toc122895083)

[1.3.8 范围证明 1](#_Toc122895084)

[1.4 参考资料 2](#_Toc122895085)

[2 总体设计 3](#_Toc122895086)

[2.1 功能 3](#_Toc122895087)

[2.2 安装 3](#_Toc122895088)

[2.3 模块结构 3](#_Toc122895089)

[3 哈希函数 5](#_Toc122895090)

[3.1 SHA256 5](#_Toc122895091)

[3.1.1 流程简述 5](#_Toc122895092)

[3.1.2 函数和接口说明 5](#_Toc122895093)

[3.2 SHA512 6](#_Toc122895094)

[3.2.1 流程简述 6](#_Toc122895095)

[3.2.2 函数和接口说明 6](#_Toc122895096)

[3.3 Keccak 7](#_Toc122895097)

[3.3.1 流程简述 7](#_Toc122895098)

[3.3.2 函数和接口说明 7](#_Toc122895099)

[3.4 SHA3 8](#_Toc122895100)

[3.4.1 流程简述 8](#_Toc122895101)

[3.4.2 函数和接口说明 8](#_Toc122895102)

[3.5 Scrypt 10](#_Toc122895103)

[3.5.1 流程简述 10](#_Toc122895104)

[3.5.2 函数和接口说明 10](#_Toc122895105)

[3.6 Poseidon 11](#_Toc122895106)

[3.6.1 流程简述 11](#_Toc122895107)

[3.6.2 函数和接口说明 11](#_Toc122895108)

[4 数字签名 12](#_Toc122895109)

[4.1 ECDSA 12](#_Toc122895110)

[4.1.1 流程简述 12](#_Toc122895111)

[4.1.2 函数和接口说明 12](#_Toc122895112)

[4.2 ECSchnorr 13](#_Toc122895113)

[4.2.1 流程简述 13](#_Toc122895114)

[4.2.2 函数和接口说明 14](#_Toc122895115)

[4.3 BLS 14](#_Toc122895116)

[4.3.1 流程简述 15](#_Toc122895117)

[4.3.2 函数和接口说明 15](#_Toc122895118)

[4.4 BLS（纯go） 16](#_Toc122895119)

[4.4.1 流程简述 16](#_Toc122895120)

[4.4.2 函数和接口说明 16](#_Toc122895121)

[4.5 EdDSA 17](#_Toc122895122)

[4.5.1 流程简述 17](#_Toc122895123)

[4.5.2 函数和接口说明 18](#_Toc122895124)

[4.6 SM2 18](#_Toc122895125)

[4.6.1 流程简述 19](#_Toc122895126)

[4.6.2 函数和接口说明 19](#_Toc122895127)

[4.7 BLS多重签名 20](#_Toc122895128)

[4.7.1 流程简述 20](#_Toc122895129)

[4.7.2 函数和接口说明 20](#_Toc122895130)

[4.8 BLS门限签名 21](#_Toc122895131)

[4.8.1 流程简述 22](#_Toc122895132)

[4.8.2 函数和接口说明 23](#_Toc122895133)

[5 密钥交换 24](#_Toc122895134)

[5.1 DH 24](#_Toc122895135)

[5.1.1 流程简述 24](#_Toc122895136)

[5.1.2 函数和接口说明 25](#_Toc122895137)

[5.2 ECDH 26](#_Toc122895138)

[5.2.1 流程简述 26](#_Toc122895139)

[5.2.2 函数和接口说明 27](#_Toc122895140)

[5.3 Stealth Address 27](#_Toc122895141)

[5.3.1 流程简述 27](#_Toc122895142)

[5.3.2 函数和接口说明 28](#_Toc122895143)

[6 秘密承诺 30](#_Toc122895144)

[6.1 哈希承诺 30](#_Toc122895145)

[6.1.1 流程简述 30](#_Toc122895146)

[6.1.2 函数和接口说明 30](#_Toc122895147)

[6.2 Pedersen承诺 31](#_Toc122895148)

[6.2.1 流程简述 31](#_Toc122895149)

[6.2.2 函数和接口说明 31](#_Toc122895150)

[6.3 多项式承诺 32](#_Toc122895151)

[6.3.1 流程简述 33](#_Toc122895152)

[6.3.2 函数和接口说明 33](#_Toc122895153)

[6.4 向量承诺 35](#_Toc122895154)

[6.4.1 流程简述 35](#_Toc122895155)

[6.4.2 函数和接口说明 35](#_Toc122895156)

[7 秘密分享 38](#_Toc122895157)

[7.1 Shamir秘密分享 38](#_Toc122895158)

[7.1.1 流程简述 38](#_Toc122895159)

[7.1.2 函数和接口说明 38](#_Toc122895160)

[7.2 可验证秘密分享 39](#_Toc122895161)

[7.2.1 流程简述 39](#_Toc122895162)

[7.2.2 函数和接口说明 39](#_Toc122895163)

[8 可验证随机数VRF 41](#_Toc122895164)

[8.1 流程简述 41](#_Toc122895165)

[8.2 函数和接口说明 41](#_Toc122895166)

[9 验证延迟函数VDF 43](#_Toc122895167)

[9.1 流程简述 43](#_Toc122895168)

[9.2 函数和接口说明 43](#_Toc122895169)

[10 零知识证明zk-SNARK 45](#_Toc122895170)

[10.1 流程简述 45](#_Toc122895171)

[10.2 函数和接口说明 45](#_Toc122895172)

[11 Bulletproofs 48](#_Toc122895173)

[11.1 流程简述 48](#_Toc122895174)

[11.2 函数和接口说明 48](#_Toc122895175)

# 引言

## 编写目的

本文档的目的是对密码学库进行详细设计说明，以便跨链系统项目开发人员了解密码学库中原语的实现与调用方法，为开发提供参考。以下叙述将结合文字描述、伪代码以及图表等来描述密码学库的详细设计和相关的模块描述。

本报告的预期读者有开发人员以及跟该项目相关的其他人员。

## 项目背景

本密码学库实现跨链系统所需密码学原语，为跨链项目的开发提供密码学原语的函数和接口调用。

## 相关术语

本部分描述了相关密码学概念，包括哈希函数、数字签名、密钥协商、秘密分享、可验证随机数、验证延迟函数、零知识证明。

### 哈希函数

### 数字签名

### 密钥交换

### 秘密承诺

### 秘密分享

### 可验证随机数

### 可验证延迟函数

### 范围证明

## 参考资料

# 总体设计

## 功能

本密码学库的功能需求如下：

（1）提供哈希函数原语；

（2）提供数字签名原语；

（3）提供密钥交换原语；

（4）提供秘密承诺原语；

（5）提供秘密分享原语；

（6）提供可验证随机数原语；

（7）提供验证延迟函数原语；

（8）提供范围证明原语；

本密码学库的非功能需求如下：

（1）安全性：所实现的原语

（2）硬件环境：支持在Linux系统环境下运行；

## 安装

在库的根目录中使用命令行执行下列指令

go mod tidy

### 配置VDF运行环境

VDF需要独占一个CPU运行，因此需要改变系统配置

GRUB\_CMDLINE\_LINUX\_DEFAULT=“isolcpus=1”

### 2.2.1 安装bn库

用处：用于支持4.3节BLS签名

安装步骤：

（1）命令行运行git clone https://github.com/dfinity/bn，下载源代码。

（2）进入bn库源代码根目录下，命令行依次运行make和make install命令，将源码编译并安装到/usr/local/lib下。

（3）命令行运行export LD\_LIBRARY\_PATH=/lib:/usr/lib:/usr/local/lib，确保链接程序找得到该目录。

## 模块结构

|  |  |  |
| --- | --- | --- |
| 模块 | 所在位置 | 负责人 |
| 哈希函数 | blockchain-crypto/hash | 丁振洋 |
| 数字签名 | blockchain-crypto/signature | 张天逸 |
| 密钥交换 | blockchain-crypto/key\_exchange | 冯仲达 |
| 秘密承诺 | blockchain-crypto/commitment | 冯仲达 |
| 秘密分享 | blockchain-crypto/secret\_sharing | 冯仲达 |
| VRF | blockchain-crypto/vrf | 冯仲达 |
| VDF | blockchain-crypto/vdf | 杨洋 |
| zk-SNARK | blockchain-crypto/zksnark/gnark | 杨洋 |
| Bulletproof | blockchain-crypto/bulletproof | 杨洋 |

# 哈希函数

本部分针对密码学库中的哈希函数模块进行详细说明。哈希函数部分实现了SHA256、Keccak256、Keccak512、SHA3、Scrypt、Poseidon哈希函数。

## 流程简述

### 无需传入额外参数

SHA256、Keccak256、Keccak512、SHA3、Poseidon

（1）导入hash包；

（2）调用Hash()，输入哈希函数标识字符串和待哈希数据，返回哈希值和错误；

（3）判断是否存在错误

### 需传入额外参数（Scrypt）

Scrypt

（1）导入hash包；

（2）调用WithArgs()，输入额外参数（如），创建Option类型；

（3）调用Hash()，输入哈希函数标识字符串、待哈希数据和Option类型的额外参数，返回哈希值；

（4）判断是否存在错误

### 各哈希函数标识字符串

|  |  |  |
| --- | --- | --- |
| 哈希函数 | 哈希值位数 | 标识字符串 |
| RIPEMD-160 | 160 | ripemd160 |
| SHA-256 | 256 | sha256 |
| SHA-512 | 512 | sha512 |
| SHA3 | 256 | sha3-256 |
| 512 | sha3-512 |
| Keccak | 256 | keccak256 |
| 512 | keccak512 |
| Poseidon | 256 | poseidon256 |
| Scrypt | — | scrypt |

## 函数和接口说明

|  |  |
| --- | --- |
| Hash(hashType string, input []byte, options ...Option) ([]byte, error) | |
| 说明 | 进行哈希函计算。 |
| 前置条件 | hash包已经被导入。 |
| 后置条件 | 需要判断函数返回是否存在错误。 |
| 参数 | hashType：要使用的哈希函数类型，string类型。  input：待哈希数据，[]byte类型  options：可选参数，Option类型 |
| 返回值 | 哈希值，[]byte类型，若存在错误则返回nil；  错误，error类型 |

|  |  |
| --- | --- |
| WithArgs(salt []byte, N, r, p, keyLen int) Option | |
| 说明 | 辅助传入额外参数。 |
| 前置条件 | hash包已经被导入。 |
| 后置条件 | 需要判断函数返回是否存在错误。 |
| 参数 | salt：盐值， []byte类型。  N：CPU/内存成本参数，必须是2的幂（大于1），[]byte类型  r：blocksize参数，int类型  p：并行参数，int类型  keyLen：哈希值的字节长度，int类型 |
| 返回值 | 额外参数函数，Option类型，用于传递额外参数； |

# 数字签名

本节将对数字签名部分进行详细说明。数字签名部分目前实现了共7种算法，分别是ECDSA、ECSchnorr、BLS、EdDSA、SM2、BLS门限签名、BLS多重签名，其中BLS有调用C和纯GO两种实现。

## 常规签名

所在包：blockchain-crypto/signature

### 流程简述

**签名者：**

（1）导入blockchain-crypto/signature包。

（2）签名者进行密钥生成，调用KeygenApi()，输入方案名称，获得公私钥对，保存私钥，将公钥发送给验证者。

（3）签名者进行签名生成，调用SignApi()，输入方案名称、私钥、消息，获得签名，将签名发送给验证者。

**验证者：**

（1）导入blockchain-crypto/signature包。

（2）验证者进行签名验证，调用VerifyApi()，输入方案名称、公钥、信息、签名，获得签名是否有效的判断结果。

### 函数和接口说明

|  |  |
| --- | --- |
| KeygenApi(schemenme string) ([]string, []string) | |
| 说明 | 密钥生成接口。 |
| 前置条件 | blockchain-crypto/signature包已经被导入。 |
| 后置条件 | 得到的密钥用于签名生成和验证。 |
| 参数 | schemenme：方案名称，string类型，有效值为bls、bls\_purego、ecdsa、ecschnorr、eddsa、sm2。 |
| 返回值 | 私钥，[]string类型。  压缩后的公钥，[]string类型。 |

|  |  |
| --- | --- |
| SignApi(schemenme string, seck []string, mes string) []string | |
| 说明 | 签名生成接口。 |
| 前置条件 | blockchain-crypto/signature包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | schemenme：方案名称，string类型，有效值为bls、bls\_purego、ecdsa、ecschnorr、eddsa、sm2。  seck：私钥，[]string类型。  mes：待签名的消息，string类型。 |
| 返回值 | 签名，[]string类型。 |

|  |  |
| --- | --- |
| VerifyApi(schemenme string, pubk []string, mes []string, sig []string) bool | |
| 说明 | 签名验证接口。 |
| 前置条件 | blockchain-crypto/signature包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | schemenme：方案名称，string类型，有效值为bls、bls\_purego、ecdsa、ecschnorr、eddsa、sm2。  pubk：公钥（会自动解析是否压缩），[]string类型。  mes：被签名的消息，string类型。  sig：签名，[]string类型。 |
| 返回值 | 签名是否有效，bool类型。 |

## BLS多重签名

所在包：blockchain-crypto/signature/bls\_multi

### 流程简述

**每个签名者分别进行：**

（1）导入blockchain-crypto/signature/bls\_multi包。

（2）签名者分别进行密钥生成，调用KeygenApi()生成公私钥对，保存私钥，将公钥发送给验证者。

（3）签名者分别进行签名生成，调用Sign\_api()，输入私钥、消息，获得签名，并将签名交给同一进行签名整合的实体（任意实体均可）。

**收到签名的实体：**

（1）导入blockchain-crypto/signature/bls\_multi包。

（2）任意收到签名的实体可以将多个签名进行整合，调用MultiSign\_api()，输入多个集合，获得多重签名，并将多重签名发送给验证者。

**验证者：**

（1）导入blockchain-crypto/signature/bls\_multi包。

（2）验证者验证多重签名，调用MultiVerify\_api()，输入公钥集合、信息、多重签名，获得签名是否有效的判断结果。

### 函数和接口说明

|  |  |
| --- | --- |
| KeygenApi() (string, string) | |
| 说明 | 密钥生成接口。 |
| 前置条件 | blockchain-crypto/signature/BLS\_Multi包已经被导入。 |
| 后置条件 | 得到的密钥用于签名生成和验证。 |
| 参数 | 无。 |
| 返回值 | 私钥，string类型。  公钥，string类型。 |

|  |  |
| --- | --- |
| Sign\_api(seck string, mes string) string | |
| 说明 | 签名生成接口。 |
| 前置条件 | blockchain-crypto/signature/BLS\_Multi包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | seck：私钥，string类型。  mes：消息，string类型。 |
| 返回值 | 签名，string类型。 |

|  |  |
| --- | --- |
| MultiSign\_api(sigs []string) string | |
| 说明 | 签名整合接口。 |
| 前置条件 | blockchain-crypto/signature/BLS\_Multi包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | sigs：签名集合，[]string类型。 |
| 返回值 | 多重签名，string类型。 |

|  |  |
| --- | --- |
| MultiVerify\_api(pubks []string, mes string, mulsig string) bool | |
| 说明 | 多重签名验证接口。 |
| 前置条件 | blockchain-crypto/signature/BLS\_Multi包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | pubks：公钥集合，[]string类型。  mes：消息，string类型。  mulsig：多重签名，string类型。 |
| 返回值 | 多重签名是否有效，bool类型。 |

## BLS门限签名

所在包：blockchain-crypto/signature/bls\_threshold

### 流程简述

**整体流程**

分发者首先调用常规的BLS密钥生成算法生成一个私钥，然后将这个私钥通过**秘密分发接口**生成一个秘密共享，包括ID、私钥份额、公钥份额，然后秘密地分发给参与者。

任意t个及以上的参与者可以生成有效的签名。他们可以先调用**私钥恢复接口**恢复私钥，再使用私钥进行常规的BLS签名；也可以先分别调用常规的BLS签名生成算法生成签名份额，再调用**签名恢复接口**恢复签名。这两种方式生成的签名值应当是相同且有效的。

验证者可以直接从最开始的分发者处获得公钥，也可以由大于等于t个持有份额的参与者调用**公钥恢复接口**恢复出公钥，然后调用常规的BLS签名验证算法进行验证。

**密钥生成及分发**

**密钥的分发者进行：**

（1）导入blockchain-crypto/signature/bls\_threshold包。

（2）进行密钥生成，调用BLS.KeygenApi()生成公私钥对，保存私钥，将公钥公开。

（3）对私钥进行秘密分享，调用SecKeyShare\_api()生成ID、私钥、公钥的份额，分别保密地交给所有参与分享的参与者。

**私钥恢复**

**部分拥有份额的参与者进行：**

（1）导入blockchain-crypto/signature/bls\_threshold包。

（2）尝试恢复私钥，调用SecKeyRecover\_api()，输入ID数组和私钥份额数组，获得恢复的私钥。

**签名恢复**

**部分拥有份额的参与者进行：**

（1）导入blockchain-crypto/signature/bls\_threshold包。

（2）尝试恢复签名，调用SigRecover\_api()，输入ID数组和签名份额数组，获得恢复的签名。

**公钥恢复**

**部分拥有份额的参与者进行：**

（1）导入blockchain-crypto/signature/bls\_threshold包。

（2）尝试恢复公钥，调用PubKeyRecover\_api()，输入ID数组和公钥数组，获得恢复的公钥。

### 函数和接口说明

|  |  |
| --- | --- |
| SecKeyShare\_api(seck string, t int, n int) ([]string, []string, []string) | |
| 说明 | 秘密分发接口。 |
| 前置条件 | blockchain-crypto/signature/bls\_threshold包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | seck：要进行秘密分享的私钥，string类型。  t：门限值，int类型。  n：总份额数，int类型。 |
| 返回值 | ID数组，[]string类型。  私钥数组，[]string类型。  公钥数组，[]string类型。 |

|  |  |
| --- | --- |
| func SecKeyRecover\_api(idVector []string, secVector []string) (bool, string) | |
| 说明 | 私钥恢复接口。 |
| 前置条件 | blockchain-crypto/signature/bls\_threshold包已经被导入。  调用者收集了数个参与者持有的私钥份额。 |
| 后置条件 | 无。 |
| 参数 | idVector：ID数组，[]string类型。  secVector：私钥份额数组，[]string类型。 |
| 返回值 | 是否正常运行，bool类型。  恢复的私钥，string类型。 |

|  |  |
| --- | --- |
| func SigRecover\_api(idVector []string, sigVector []string) (bool, string) | |
| 说明 | 签名恢复接口。 |
| 前置条件 | blockchain-crypto/signature/bls\_threshold包已经被导入。  数个签名者使用其持有的私钥份额进行了签名，调用者收集了这些签名份额。 |
| 后置条件 | 无。 |
| 参数 | idVector：ID数组，[]string类型。  sigVector：签名份额数组，[]string类型。 |
| 返回值 | 是否正常运行，bool类型。  恢复的签名，string类型。 |

|  |  |
| --- | --- |
| func PubKeyRecover\_api(idVector []string, pubVector []string) (bool, string) | |
| 说明 | 公钥恢复接口。 |
| 前置条件 | blockchain-crypto/signature/bls\_threshold包已经被导入。  调用者收集了数个参与者持有私钥对应的公钥份额。 |
| 后置条件 | 无。 |
| 参数 | idVector：ID数组，[]string类型。  pubVector：公钥份额数组，[]string类型。 |
| 返回值 | 是否正常运行，bool类型。  恢复的公钥，string类型。 |

# 密钥交换

本部分针对密码学库中的密钥交换协议模块进行详细说明。秘密交换协议部分实现Ddiffie-Hellman密钥交换协议（DH），椭圆曲线上的Ddiffie-Hellman密钥交换协议（Elliptic Curve Diffie-Hellman，ECDH），隐地址算法（Stealth address）。

## DH

所在包：blockchain-crypto/KeyExchange/DH

### 流程简述

**会话者a**

（1）导入DH包。

（2）给定群g，调用GenerateKey()函数获得私钥private\_a，公钥public\_a。

（3）给定私钥private\_a，公钥public\_b，调用ComputeSecret()函数获得会话密钥secret。

**会话者b**

（1）导入DH包。

（2）给定群g，调用CalculateKeypair()函数获得私private\_b，公钥public\_b。

（3）给定私钥private\_b，公钥public\_a，调用ComputeSecret()函数获得会话密钥secret。

### 函数和接口说明

|  |  |
| --- | --- |
| Init\_2048() (g \*Group) | |
| 说明 | 群生成函数。 |
| 前置条件 | DH包已经被导入。 |
| 后置条件 | 得到的群g将用于公私钥对的生成和会话密钥的生成。 |
| 参数 | 无 |
| 返回值 | g，包含群的信息（群所模素数和生成元，其中素数为2048位）， \*Group类型。 |

|  |  |
| --- | --- |
| Init\_3072() (g \*Group) | |
| 说明 | 群生成函数。 |
| 前置条件 | DH包已经被导入。 |
| 后置条件 | 得到的群g将用于公私钥对的生成和会话密钥的生成。 |
| 参数 | 无 |
| 返回值 | g，包含群的信息（群所模素数和生成元，其中素数为3072位） \*Group类型。 |

|  |  |
| --- | --- |
| Init\_4096() (g \*Group) | |
| 说明 | 群生成函数。 |
| 前置条件 | DH包已经被导入。 |
| 后置条件 | 得到的群g将用于公私钥对的生成和会话密钥的生成。 |
| 参数 | 无 |
| 返回值 | g，包含群的信息（群所模素数和生成元，其中素数为4096位） \*Group类型。 |

|  |  |
| --- | --- |
| (g \*Group) GenerateKey() (private PrivateKey, public PublicKey, err error) | |
| 说明 | 公私钥对生成函数。 |
| 前置条件 | DH包已经被导入。 |
| 后置条件 | 得到的私钥private和公钥public将用于会话密钥的生成， |
| 参数 | 无 |
| 返回值 | private，私钥，PrivateKey类型。  public，公钥， PublicKey类型  err，错误信息（如g未赋值） error类型。 |

|  |  |
| --- | --- |
| (g \*Group) ComputeSecret(private PrivateKey, peersPublic PublicKey) (secret \*big.Int) | |
| 说明 | 公私钥对生成函数。 |
| 前置条件 | DH包已经被导入。 |
| 后置条件 | 无 |
| 参数 | private，私钥，PrivateKey类型。  peersPublic，其他会话者的公钥，PublicKey类型 |
| 返回值 | secret，会话密钥，\*big.Int类型。 |

## ECDH

所在包：blockchain-crypto/KeyExchange/ECDH

### 流程简述

**会话者a**

（1）导入ECDH包。

（2）调用CalculateKeypair()函数获得私钥priv\_a，公钥pub\_a。

（3）给定私钥priv\_a，公钥pub\_b，调用CalculateNegotiationKey()函数获得会话密钥shared。

**会话者b**

（1）导入ECDH包。

（2）调用CalculateKeypair()函数获得私钥priv\_b，公钥pub\_b。

（3）给定私钥priv\_b，公钥pub\_a，调用CalculateNegotiationKey()函数获得会话密钥shared。

### 函数和接口说明

|  |  |
| --- | --- |
| CalculateKeypair() (priv \*big.Int, pub ecdsa.PublicKey) | |
| 说明 | 公私钥对生成函数（在椭圆曲线p256上）。 |
| 前置条件 | ECDH包已经被导入。 |
| 后置条件 | 得到的私钥priv和公钥将用于会话密钥的生成。 |
| 参数 | 无 |
| 返回值 | priv，私钥， \*big.Int类型。  pub，公钥，ecdsa.PublicKey类型。 |

|  |  |
| --- | --- |
| CalculateNegotiationKey(pub ecdsa.PublicKey, priv \*big.Int) (shared [32]byte) | |
| 说明 | 会话密钥生成函数。 |
| 前置条件 | ECDH包已经被导入。 |
| 后置条件 | 无 |
| 参数 | priv，私钥， \*big.Int类型。  pub，公钥，ecdsa.PublicKey类型。 |
| 返回值 | shared，会话密钥， [32]byte类型。 |

## Stealth Address

所在包：blockchain-crypto/KeyExchange/Stealthaddress

### 流程简述

**接收方**

1. 导入Stealthaddress包。
2. 给定随机字符串，调用RecCalculateKeypairs()函数生成接收方公私钥对priv1，pub1，priv2，pub2。

**发送方**

（3）导入Stealthaddress包.

（4）给定私钥r，接收方公钥pub1，pub2，调用SendCalculateObfuscateAddress()函数计算隐地址P。

（5）给定私钥r，调用SendCalculatePublicKey()函数计算公钥R。

**接收方**

1. 给定隐地址P，对应的公钥R，自身的私钥priv1，调用RecCalculateObfuscateAddress()函数验证交易接收方是否为自身。

（7）给定经RecCalculateObfuscateAddress ()函数验证通过的公钥R，利用自身的私钥priv1，priv2调用RecCalculateAddressPrivatekey()函数计算隐地址对应的私钥priv。

### 函数和接口说明

|  |  |
| --- | --- |
| RecCalculateKeypairs(str1,str2 string) (priv1 \*big.Int, pub1 \*PublicKey, priv2 \*big.Int, pub2 \*PublicKey) | |
| 说明 | 公私钥对生成函数（在椭圆曲线p256上）。 |
| 前置条件 | Stealthaddress包已经被导入。 |
| 后置条件 | 无 |
| 参数 | str1，随机字符串用于计算私钥，string类型。  str2，随机字符串用于计算私钥，string类型。 |
| 返回值 | priv1，私钥，\*big.Int类型。  pub1，公钥，\*PublicKey类型。  priv2，私钥， \*big.Int类型。  pub2，公钥， \*PublicKey类型。 |

|  |  |
| --- | --- |
| SendCalculateObfuscateAddress(r \*big.Int,pub1,pub2 \*PublicKey)(P \*PublicKey) | |
| 说明 | 隐地址计算函数。 |
| 前置条件 | Stealthaddress包已经被导入。 |
| 后置条件 | 无 |
| 参数 | r ，私钥，\*big.Int类型。  pub1，公钥，\*PublicKey类型。  pub2，公钥，\*PublicKey类型。 |
| 返回值 | P，隐地址，\*PublicKey类型。 |

|  |  |
| --- | --- |
| SendCalculatePublicKey(r \*big.Int)(R \*PublicKey) | |
| 说明 | 公钥生成函数。 |
| 前置条件 | Stealthaddress包已经被导入。 |
| 后置条件 | 无 |
| 参数 | r ，私钥，\*big.Int类型。 |
| 返回值 | R，公钥，\*PublicKey类型。 |

|  |  |
| --- | --- |
| RecCalculateObfuscateAddress(P, R \*PublicKey, priv1 \*big.Int,pub2 \*PublicKey)(bool) | |
| 说明 | 隐地址验证函数。 |
| 前置条件 | Stealthaddress包已经被导入。 |
| 后置条件 | 无 |
| 参数 | 1. 隐地址，\*PublicKey类型。   R，公钥，\*PublicKey类型。  priv1，私钥，\*big.Int类型。  pub2，公钥，\*PublicKey类型。 |
| 返回值 | true表示该隐地址对应的交易的接收方为私钥priv1的持有者，false则反之，bool类型。 |

|  |  |
| --- | --- |
| RecCalculateAddressPrivatekey(priv1,priv2 \*big.Int, R \*PublicKey) (priv \* big.Int) | |
| 说明 | 隐地址对应的私钥生成函数。 |
| 前置条件 | Stealthaddress包已经被导入。 |
| 后置条件 | 无 |
| 参数 | priv1，私钥，\*big.Int类型。  priv2，私钥，\*big.Int类型。  R，公钥，\*PublicKey类型。 |
| 返回值 | priv，私钥，\*big.Int类型。 |

# 秘密承诺

本部分针对密码学库中的秘密承诺协议模块进行详细说明。秘密承诺协议部分实现了哈希承诺协议（Hash Commitment），Pedersen承诺协议（Pedersen Commitment），多项式承诺协议（Polynomial commitment），向量承诺协议（Vector Commitment）。

## 哈希承诺

所在包：blockchain-crypto/Commitment/HashCommitment

### 流程简述

**承诺者**

（1）导入HashCommitment包。

（2）给定需要承诺的字符串s，随机字符串r，调用Commit()函数获得承诺值commit。

**验证者**

（1）导入HashCommitment包。

（2）给定承诺的字符串s，和承诺字符串对应的字符串r，承诺值commit，调用Verify()函数验证承诺字符串的有效性。

### 函数和接口说明

|  |  |
| --- | --- |
| Commit(s,r string) (commit [32]byte) | |
| 说明 | 承诺生成函数。 |
| 前置条件 | HashCommitment包已经被导入。 |
| 后置条件 | 得到的承诺值commit将用于对承诺字符串的验证。 |
| 参数 | s，需要承诺的字符串，string类型。  r，随机字符串，string类型。 |
| 返回值 | commit，承诺值，[32]byte类型。 |

|  |  |
| --- | --- |
| Verify(s,r string, commit [32]byte) | |
| 说明 | 验证承诺函数 |
| 前置条件 | HashCommitment包已经被导入 |
| 后置条件 | 无。 |
| 参数 | s，承诺的字符串，string类型。  r，随机字符串，string类型。  commit，承诺值，[32]byte类型。 |
| 返回值 | true表示承诺的字符串和承诺值一致，false表示承诺的字符串和承诺值不一致，bool类型。 |

## Pedersen承诺

所在包：blockchain-crypto/Commitment/PedersenCommitment

### 流程简述

**承诺者**

（1）导入PedersenCommitmentr包。

（2）调用ParamsGen() 函数获得随机生成元G、H。

（3）调用RandomGen()函数获得随机因子r。

（4）给定随机生成元G、H，要承诺的值s，随机因子r，调用Commit()函数获得承诺值commit。

**验证者**

（1）导入PedersenCommitmentr包。

（2）给定承诺值commit，随机生成元G、H，要承诺的值s，与要承诺的值对应的随机因子r，调用Verify()函数验证要承诺的值的有效性。

### 函数和接口说明

|  |  |
| --- | --- |
| ParamsGen() (G, H ristretto.Point) | |
| 说明 | 随机生成元生成函数。 |
| 前置条件 | PedersenCommitmentr包已经被导入。 |
| 后置条件 | 得到的生成元G、H将用于承诺值的生成和验证。 |
| 参数 | 无。 |
| 返回值 | G，随机生成元，ristretto.Point类型。  H，随机生成元，ristretto.Point类型。 |

|  |  |
| --- | --- |
| RandomGen() (r ristretto.Scalar) | |
| 说明 | 随机因子生成函数。 |
| 前置条件 | PedersenCommitmentr包已经被导入。 |
| 后置条件 | 得到的随机因子r将用于承诺值的生成和验证。 |
| 参数 | 无。 |
| 返回值 | r，随机因子， ristretto.Scalar类型。 |

|  |  |
| --- | --- |
| func Commit(G, H ristretto.Point, s uint64, r ristretto.Scalar) (commit ristretto.Point) | |
| 说明 | 承诺生成函数。 |
| 前置条件 | PedersenCommitmentr包已经被导入。 |
| 后置条件 | 得到的承诺值commit将用于对要承诺的值的验证。 |
| 参数 | G，随机生成元，ristretto.Point类型。  H，随机生成元，ristretto.Point类型。  s，要承诺的值，uint64类型。  r，随机因子， ristretto.Scalar类型。 |
| 返回值 | commit，承诺值，ristretto.Point类型。 |

|  |  |
| --- | --- |
| func Verify(commit, G, H ristretto.Point, s uint64, r ristretto.Scalar) bool | |
| 说明 | 承诺验证函数。 |
| 前置条件 | PedersenCommitmentr包已经被导入。 |
| 后置条件 | 无 |
| 参数 | commit，承诺值。  G，随机生成元，ristretto.Point类型。  H，随机生成元，ristretto.Point类型。  s，要承诺的值，uint64类型。  r，与要承诺的值对应的随机因子， ristretto.Scalar类型。 |
| 返回值 | true表示要承诺的值和承诺值一致，false表示要承诺的值和承诺值不一致，bool类型。 |

## 多项式承诺

所在包：blockchain-crypto/Commitment/PolyCommitment

### 流程简述

可信机构

（1）导入PolyCommitment包。

（2）给定随机字符串str，可以承诺的多项式的最大维度n，调用Init()函数获得秘密参数c，公开参数信息T。

承诺者

1. 导入PolyCommitment包。
2. 给定要承诺的多项式系数的值的切片p，公开参数信息T，调用Commit()函数获得承诺值r。
3. 给定选取的公开参数信息T，要承诺的多项式系数的值的切片p，对多项式进行评估的位置d，调用CreatWitness()函数获得对多项式进行评估的位置X，对多项式进行评估的值Y，对多项式评估的见证W。

验证者

（1）导入PolyCommitment包。

（2）给定选取的公开参数信息T，承诺值r，对多项式进行评估的位置X，对多项式进行评估的值Y，对多项式评估的见证W，调用VerifyWitness()函数验证评估的有效性。

### 函数和接口说明

|  |  |
| --- | --- |
| Init(str string, n int) (c []byte, T \*Trust) | |
| 说明 | 参数生成函数。 |
| 前置条件 | PolyCommitment包已经被导入。 |
| 后置条件 | 得到的公开参数信息T将用于承诺值的生成和验证，见证的生成和验证。 |
| 参数 | str ，随机字符串，string类型。  n，可以承诺的多项式的最大维度， int类型。 |
| 返回值 | c，选取的秘密参数， []byte类型。  T ，选取的公开参数信息（，），\*Trust类型。 |

|  |  |
| --- | --- |
| Commit(p []\*big.Int, T \*Trust) (r \*bn256.G1) | |
| 说明 | 承诺生成函数。 |
| 前置条件 | PolyCommitment包已经被导入。 |
| 后置条件 | 得到的承诺值r将用于承诺值的验证和见证的验证。 |
| 参数 | p，要承诺的多项式系数的值的切片，[]\*big.Int类型。  T，选取的公开参数信息，\*Trust类型。 |
| 返回值 | r，承诺值， \*bn256.G1类型。 |

|  |  |
| --- | --- |
| Verify(r \*bn256.G1, p []\*big.Int, T \*Trust) bool | |
| 说明 | 签名验证接口。 |
| 前置条件 | PolyCommitment包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | r，承诺值， \*bn256.G1类型。  p，承诺的多项式系数的值的切片，[]\*big.Int类型。  T，选取的公开参数信息，\*Trust类型。 |
| 返回值 | true表示要承诺的多项式的系数的值和承诺值一致，false表示要承诺的多项式的系数的值和承诺值不一致，bool类型。 |

|  |  |
| --- | --- |
| CreatWitness(T \*Trust, p []\*big.Int, d \*big.Int) (X, Y \*big.Int, W \*bn256.G1) | |
| 说明 | 见证生成函数。 |
| 前置条件 | PolyCommitment包已经被导入。 |
| 后置条件 | 得到的对多项式评估X，Y和评估的见证将被用于见证的验证 |
| 参数 | T，选取的公开参数信息，\*Trust类型。  p，要承诺的多项式系数的值的切片，[]\*big.Int类型。  d ，对多项式进行评估的位置（即y=f(x)中的x值），\*big.Int类型。 |
| 返回值 | X， 对多项式进行评估的位置（即y=f(x)中的x值），\*big.Int类型。  Y， 对多项式进行评估的值（即y=f(x)中的y值），\*big.Int类型。  W，对多项式评估的见证，\*bn256.G1类型 |

|  |  |
| --- | --- |
| VerifyWitness(T \*Trust, r \*bn256.G1, X, Y \*big.Int, W \*bn256.G1) bool | |
| 说明 | 见证验证函数。 |
| 前置条件 | PolyCommitment包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | 1. 选取的公开参数信息，\*Trust类型。   r，承诺值， \*bn256.G1类型。  X， 对多项式进行评估的位置，\*big.Int类型。  Y， 对多项式进行评估的值，\*big.Int类型。  W，对多项式评估的见证，\*bn256.G1类型 |
| 返回值 | true表示要评估正确，false表示评估错误，bool类型。 |

## 向量承诺

所在包：blockchain-crypto/Commitment/VectorCommitment

### 流程简述

**可信机构**

（1）导入VectorCommitment包。

（2）给定随机字符串str，可以承诺的向量的最大维度n，调用Init()函数获得秘密参数c，公开参数信息T。

**承诺者**

（1）导入VectorCommitment包。

（2）给定要承诺的向量每一维度的值的切片p，公开参数信息T，调用Commit()函数获得承诺值r。

（3）给定选取的公开参数信息T，要承诺的向量每一维度的值的切片p，向量的维度i，调用CreatWitness()函数获得向量在i处的评估值I，对评估的见证W。

**验证者**

（1）导入VectorCommitment包。

（2）给定选取的公开参数信息T，承诺值r，向量的维度i，向量在i处的评估值I，对评估的见证W，调用VerifyWitness()函数验证评估的有效性。

### 函数和接口说明

|  |  |
| --- | --- |
| Init(str string, n int) (c []byte, T \*Trust) | |
| 说明 | 参数生成函数。 |
| 前置条件 | VectorCommitment包已经被导入。 |
| 后置条件 | 得到的公开参数信息T将用于承诺值的生成，见证的生成和验证。 |
| 参数 | str ，随机字符串，string类型。  n，可以承诺的多项式的最大维度， int类型。 |
| 返回值 | c，选取的秘密参数， []byte类型。  T ，选取的公开参数信息（），\*Trust类型。。 |

|  |  |
| --- | --- |
| Commit(p []\*big.Int, T \*Trust) (r \*bn256.G1) | |
| 说明 | 承诺生成函数。 |
| 前置条件 | VectorCommitment包已经被导入。 |
| 后置条件 | 得到的承诺值r将用于见证的验证。 |
| 参数 | p，要承诺的向量每一维度的值的切片，[]\*big.Int类型。  T，选取的公开参数信息，\*Trust类型。 |
| 返回值 | r，承诺值， \*bn256.G1类型。 |

|  |  |
| --- | --- |
| CreatWitness(i int, p []\*big.Int, T \*Trust) (I \*big.Int, W \*bn256.G1) | |
| 说明 | 见证生成函数。 |
| 前置条件 | VectorCommitment包已经被导入。 |
| 后置条件 | 得到向量第i维的值I和见证W将被用于见证的验证。 |
| 参数 | i，向量的维度，int类型。  T，选取的公开参数信息，\*Trust类型。  p，要承诺的向量每一维度的值的切片，[]\*big.Int类型。 |
| 返回值 | I， 向量在i处的评估值，\*big.Int类型。  W，对评估的见证，\*bn256.G1类型。 |

|  |  |
| --- | --- |
| VerifyWitness(i int, I \*big.Int, T \*Trust, W \*bn256.G1, r \*bn256.G1) bool | |
| 说明 | 见证生成函数。 |
| 前置条件 | VectorCommitment包已经被导入。 |
| 后置条件 | 无 |
| 参数 | 1. 向量的维度，int类型。   I， 向量在i处的评估值，\*big.Int类型。  T，选取的公开参数信息，\*Trust类型。  W，对评估的见证，\*bn256.G1类型。  r，承诺值， \*bn256.G1类型。 |
| 返回值 | true表示要评估正确，false表示评估错误，bool类型。 |

# 秘密分享

本部分针对密码学库中的秘密分享协议模块进行详细说明。秘密分享协议部分实现了基本的秘密分享协议（Secret Sharing，SS），可验证秘密分享协议（Verifiable Secret Sharing，VSS），可公开验证秘密分享协议（Publicly Verifiable Secret Sharing，PVSS）。

## Shamir秘密分享

所在包：blockchain-crypto/SecretSharing/SS

### 流程简述

**分发者：**

（1）导入SS包；

（2）给定门限值t，秘密分享者数量n和随机字符串msg，调用Split()函数获得秘密碎片的切片shares，秘密分享的形式信息scheme，秘密值secret。

**参与者：**

（1）导入SS包；

（2）给定秘密分享的形式信息scheme，超过门限值数量的秘密碎片share，调用Recovery()恢复秘密值secret。

### 函数和接口说明

|  |  |
| --- | --- |
| Split(t, n uint32, msg string) (shares []\*sharing.ShamirShare, scheme \*sharing.Shamir, secret []byte) | |
| 说明 | 秘密分发函数。 |
| 前置条件 | SS包已经被导入。 |
| 后置条件 | 得到的秘密碎片将用于秘密值的恢复。 |
| 参数 | t：门限值，uint32类型。  n，秘密分享者数，uint32类型。  msg：用于生成秘密值的随机字符串，string类型。 |
| 返回值 | shares：秘密碎片的切片，[]\*sharing.ShamirShare类型。  scheme：秘密分享的形式信息（包含门限值以及所选取的椭圆曲线等信息），\*sharing.Shamir类型。  secret：秘密值， []byte类型。 |

|  |  |
| --- | --- |
| Recovery(scheme \*sharing.Shamir, share ...\*sharing.ShamirShare) | |
| 说明 | 秘密恢复函数。 |
| 前置条件 | SS包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | scheme：秘密分享的形式信息，\*sharing.Shamir类型。  share：秘密碎片，\*sharing.ShamirShare, |
| 返回值 | secret：秘密值， []byte类型。 |

## 可验证秘密分享

所在包：blockchain-crypto/SecretSharing/VSS

### 流程简述

**分发者：**

（1）导入VSS包；

（2）给定门限值t，秘密分享者数量n和随机字符串msg，调用Split()函数获得秘密分享多项式系数的承诺信息verifiers，秘密碎片的切片shares，秘密分享的形式信息scheme，秘密值secret。

**参与者：**

（1）导入VSS包；

（2）给定秘密分享多项式系数的承诺信息verifiers，秘密碎片share，调用Verify()函数验证秘密碎片的有效性。

（3）给定秘密分享的形式信息scheme，超过门限值数量的秘密碎片share，调用Recovery()恢复秘密值secret。

### 函数和接口说明

|  |  |
| --- | --- |
| Split(t,n uint32, msg string)(verifiers \*sharing.FeldmanVerifier,shares []\*sharing.ShamirShare,scheme \*sharing.Feldman,secret []byte) | |
| 说明 | 秘密分发函数。 |
| 前置条件 | VSS包已经被导入。 |
| 后置条件 | 得到的秘密分享多项式系数的承诺信息将用于对秘密碎片的验证，得到的秘密碎片将用于秘密值的恢复。 |
| 参数 | t：门限值，uint32类型。  n，秘密分享者数，uint32类型。  msg：用于生成秘密值的随机字符串，string类型。 |
| 返回值 | verifiers：对所构造的秘密分享多项式系数的承诺信息，\*sharing.FeldmanVerifier类型。  shares：秘密碎片的切片， []\*sharing.ShamirShare类型。  scheme： 秘密分享的形式信息（包含门限值以及所选取的椭圆曲线等信息），\*sharing.Feldman类型。  secret： 秘密值，[]byte类型。 |

|  |  |
| --- | --- |
| Verify(verifiers \*sharing.FeldmanVerifier, share \*sharing.ShamirShare)(bool) | |
| 说明 | 秘密碎片验证函数。 |
| 前置条件 | VSS包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | verifiers，对所构造的秘密分享多项式系数的承诺信息，\*sharing.FeldmanVerifier类型。  share，秘密碎片， \*sharing.ShamirShare类型。 |
| 返回值 | true表示秘密碎片在所构造的多项式上，false表示秘密碎片不在所构造的多项式上，bool类型。 |

|  |  |
| --- | --- |
| Recovery(scheme \*sharing.Feldman,share ...\*sharing.ShamirShare)(secret []byte) | |
| 说明 | 秘密恢复函数。 |
| 前置条件 | VSS包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | scheme：秘密分享的形式信息，\*sharing.Feldman类型。  shares：秘密碎片， \*sharing.ShamirShare类型。 |
| 返回值 | secret：秘密值， []byte类型。 |

# 可验证随机数VRF

本部分针对密码学库中的可验证随机函数模块进行详细说明。验证延迟函数采用google的VDF，针对一个输入，计算者利用私钥可以获得一个256位的随机输出以及证明，其他人可利用公钥验证输出的的正确性。

所在包：blockchain-crypto/VRF

## 流程简述

**计算者：**

1. 导入VRF包。
2. 调用CalculateKeypair()函数获得在椭圆曲线p256上的私钥priv和公钥pub。
3. 给定私钥priv，输入m，调用CalculateProof()函数获得256位散列值index和证据proof。

**验证者：**

（1）导入VRF包。

（2）给定散列值index，证据proof，公钥pub，调用 VerifyProof()函数验证散列值index的正确性。

## 函数和接口说明

|  |  |
| --- | --- |
| CalculateKeypair() (priv vrf.PrivateKey, pub vrf.PublicKey) | |
| 说明 | 公私钥对生成函数（p256椭圆曲线上）。 |
| 前置条件 | VRF包已经被导入。 |
| 后置条件 | 得到的私钥priv用于生成证据，得到的公钥将用于验证证据 |
| 参数 | 无。 |
| 返回值 | priv，私钥，vrf.PrivateKey类型。  pub，公钥，vrf.PublicKey类型。 |

|  |  |
| --- | --- |
| CalculateProof(priv vrf.PrivateKey, m []byte) (index [32]byte, proof []byte) | |
| 说明 | 随机数和证明生成函数。 |
| 前置条件 | VRF包已经被导入。 |
| 后置条件 | 得到的随机数index和证据proof将用于验证。 |
| 参数 | priv，私钥， vrf.PrivateKey。  m ，输入 []byte。 |
| 返回值 | index，随机数， [32]byte。  proof，证据，[]byte。 |

|  |  |
| --- | --- |
| VerifyProof(index [32]byte,proof []byte, pub vrf.PublicKey, m []byte)(bool) | |
| 说明 | 随机数验证函数。 |
| 前置条件 | VRF包已经被导入。 |
| 后置条件 | 无。 |
| 参数 | index，随机数， [32]byte。  proof，证据，[]byte。  pub，公钥，vrf.PublicKey。  m，输入，[]byte。 |
| 返回值 | true表示随机数验证通过，false表示随机数验证不通过，bool值类型。 |

# 验证延迟函数VDF

本部分针对密码学库中的验证延迟函数模块进行详细说明。验证延迟函数采用Wesolowski的VDF，针对一个输入，可以连续计算得到输出和一个验证输出的证明，计算结果不可以并行加速，利用证明可以快速验证计算的正确性。

所在包：blockchain-crypto/vdf

## 流程简述

**计算者：**

（1）导入vdf包；

（2）给定输入，调用New()初始化vdf实例结构；

（3）调用GetOutputChannel()函数定义结果的管道；

（3）调用Execute()函数执行vdf运算，输出结果和证明；

**验证者：**

（1）导入vdf包；

（2）给定输入，调用New()初始化vdf实例结构，即双方商定好vdf难度值（计算次数）和输入；

（3）根据收到的计算结果和证明调用Verify()函数对vdf的运算结果进行验证。

## 函数和接口说明

|  |  |
| --- | --- |
| New(difficulty int, input [32]byte) \*VDF | |
| 说明 | 初始化vdf实例结构中的难度和输入。 |
| 前置条件 | vdf包已经被导入。 |
| 后置条件 | 初始化的vdf实例应被用于vdf计算。 |
| 参数 | difficulty：计算次数难度值，int类型  input：输入数据，[32]byte类型。 |
| 返回值 | VDF struct 结构体。 |

|  |  |
| --- | --- |
| (vdf \*VDF) GetOutputChannel() chan [516]byte | |
| 说明 | 返回vdf输出的管道。 |
| 前置条件 | 已经初始化vdf实例结构的难度和输入。 |
| 后置条件 | 结构中的管道用于后续结果存放。 |
| 参数 | 无。 |
| 返回值 | vdf.outputChan：输出VDF结构体中的管道，用于存放VDF的结果和证明，各258bytes，chan [516]byte类型。 |

|  |  |
| --- | --- |
| (vdf \*VDF) Execute() | |
| 说明 | 执行vdf的运算过程，赋值结构体中的运算结果和证明。 |
| 前置条件 | 已经初始化并获得vdf实例。 |
| 后置条件 | 输出结果和证明用于验证。 |
| 参数 | 无。 |
| 返回值 | 无。 |

|  |  |
| --- | --- |
| (vdf \*VDF) Verify(proof [516]byte) bool | |
| 说明 | 执行vdf的验证过程 |
| 前置条件 | 已经获得vdf的输出结果与证明。 |
| 后置条件 | 输出结果和证明用于验证。 |
| 参数 | proof：证明，[516]byte类型。 |
| 返回值 | 返回是否验证通过，bool值类型。 |

# 零知识证明zk-SNARK

本部分针对密码学库中的zk-SNARK零知识证明模块进行详细说明。该模块采用官方gnark库，文档配有四个演示demo，分别为多项式方程证明、指数方程证明、哈希函数证明、merkel树证明四种零知识证明演示。Gnark官方库的github地址为https://github.com/ConsenSys/gnark，其中提供的官方技术文档地址为：https://docs.gnark.consensys.net/en/latest/。

## 流程简述

**证明者：**

（1）导入gnark包；

（2）定义电路结构体，初始化电路实例，包含公开和私有变量；

（3）调用Define()函数声明电路的约束，即公开和私有变量之间的关系，例如方程关系，哈希关系等；

（4）调用Compile()函数将电路编译成R1CS；

（5）调用Setup()函数对上述R1CS生成随机的pk和vk，将pk给Prover，将vk给Verifier；

（6）初始化witness电路结构体，即对电路赋值上公开和私有变量数值，例如方程的x和y的值，或者哈希的象和原象；

（7）调用Prove()函数计算出证明的证据；

**验证者：**

（1）导入gnark包；

（2）定义电路结构体，初始化电路实例，包含公开和私有变量；

（3）初始化publicWitness电路结构体，即只服对电路赋值上公开变量数值，例如方程的y的值，或者哈希的象；

（4）利用收到的证明和vk调用Verify()函数验证证据的真实性。

## 函数和接口说明

|  |  |
| --- | --- |
| (circuit \*Circuit) Define(curveID ecc.ID, api frontend.API) | |
| 说明 | 声明电路约束情况，定义公开和私有变量之间的运算关系（比如方程、哈希等）。 |
| 前置条件 | 电路结构体已经被定义。 |
| 后置条件 | 电路约束用于转换为R1CS。 |
| 参数 | curveID ecc.ID, api frontend.API：都为固定参数。 |
| 返回值 | 空。 |

|  |  |
| --- | --- |
| Compile(ecc.BN254, backend.GROTH16, &cubicCircuit) | |
| 说明 | 将电路编译成为R1CS。 |
| 前置条件 | 已经初始化电路结构及约束关系。 |
| 后置条件 | R1CS用于生成随机的pk和vk。 |
| 参数 | cubicCircuit：新定义的电路结构体。  ecc.BN254, backend.GROTH16：都为常数参数。 |
| 返回值 | ccs：名为CompiledConstraintSystem的interface类型，代表电路的R1CS，在example里赋值给r1cs。  err：错误，error类型。 |

|  |  |
| --- | --- |
| Setup(r1cs frontend.CompiledConstraintSystem) (ProvingKey, VerifyingKey, error) | |
| 说明 | 对上述R1CS生成随机的pk和vk。 |
| 前置条件 | 已经将电路编译为R1CS。 |
| 后置条件 | 将pk给Prover，将vk给Verifier，用于证明和验证。 |
| 参数 | r1cs：输入的R1CS，CompiledConstraintSystem的interface类型。 |
| 返回值 | ProvingKey：Groth16的证明密钥，interface类型。  VerifyingKey：Groth16的验证密钥，interface类型。  err：错误，error类型。 |

|  |  |
| --- | --- |
| Prove(r1cs frontend.CompiledConstraintSystem, pk ProvingKey, witness frontend.Circuit, opts ...func(opt \*backend.ProverOption) error) (Proof, error) | |
| 说明 | Prover生成零知识证明的证明。 |
| 前置条件 | Prover已经得到Groth16的证明密钥pk。 |
| 后置条件 | Verifier根据证明进行验证。 |
| 参数 | r1cs：输入的R1CS，CompiledConstraintSystem的interface类型。  ProvingKey：Groth16的证明密钥，interface类型。  witness：赋值上公开和私有变量数值的电路结构体。 |
| 返回值 | Proof：Prover的证明，interface类型。  err：错误，error类型。 |

|  |  |
| --- | --- |
| Verify(proof Proof, vk VerifyingKey, publicWitness frontend.Circuit) error | |
| 说明 | Verifier进行零知识证明的验证。 |
| 前置条件 | Verifier已经收到证明Proof和vk验证密钥。 |
| 后置条件 | 无。 |
| 参数 | Proof：Prover的证明，interface类型。  VerifyingKey：Groth16的验证密钥，interface类型。  publicWitness：只赋值上公开变量数值的电路结构体。 |
| 返回值 | err：错误，error类型。 |

# Bulletproofs

本部分针对密码学库中的bulletproofs模块进行详细说明。bulletproofs模块实现了bulletproofs中的范围证明，包括NewProver，commit，CreateRangeProof，Verify等零知识证明所需函数，证明特定值在范围内。

所在包：blockchain-crypto/bulletproofs

## 流程简述

**证明者：**

（1）导入Bulletproofs包；

（2）读取要证明范围的值 v 以及范围 range；

（2）调用NewProver ()初始化prover实例，并传入范围，生成数据构建实例；

（3）调用 Commit() 通过在椭圆曲线运算中加入盲因子对值 v生成承诺值；

（4）调用prover.CreateRangeProof() 函数进行零知识证明的生成运算，获取返回的证明；

**验证者：**

（1）导入Bulletproofs包；

（2）调用NewProver ()初始化prover实例，并传入范围，生成数据构建实例；

（3）调用prover.Verify() 通过收到的承诺值以及证明进行正确性验证。

## 函数和接口说明

|  |  |
| --- | --- |
| NewProver(n int) \*Prover | |
| 说明 | 创建Prover，并写入范围n。 |
| 前置条件 | 导入包，并选定好范围n。 |
| 后置条件 | 初始化的Prover后续被用于生成证明。 |
| 参数 | n：要证明的满足的范围上限。 |
| 返回值 | \*Prover：Prover 指针。 |

|  |  |
| --- | --- |
| Commit(gamma \*big.Int, h \*Point, v \*big.Int, g \*Point) \*Point | |
| 说明 | Prover 对值b 以及初始化得到的范围进行承诺 |
| 前置条件 | 已经完成范围和值的初始化。 |
| 后置条件 | 生成的承诺后续用于生成证明。 |
| 参数 | gamma：盲因子，数据类型\*bigInt。  h：椭圆曲线点，数据类型\*Point。  v：待证明的值，数据类型为\*big.Int。  g：椭圆曲线点，数据类型 \*Point。 |
| 返回值 | 返回承诺值指针。 |

|  |  |
| --- | --- |
| (v \*Prover) CreateRangeProof(V \*Point, value, gamma \*big.Int,  nonce [32]byte, message [16]byte) (BulletProof, error) | |
| 说明 | Prover 根据证明以及随机数等生成范围证明。 |
| 前置条件 | 已初始化并获得Prover 实例，并已生成承诺。 |
| 后置条件 | 生成的承诺证明后续可以进行验证。 |
| 参数 | V：承诺制, 数据类型 \*Point。  value：要证明的值，数据类型 \*bigInt。  gamma：盲因子，数据类型 \*bigInt。  nonce：随机数，[32]byte类型。  message：附加消息，[16]byte。 |
| 返回值 | 返回范围证明的证明凭证bulletproof结构体。 |

|  |  |
| --- | --- |
| (v \*Prover) Verify(V \*Point, proof BulletProof) bool | |
| 说明 | 根据承诺值以及证明凭证进行验证。 |
| 前置条件 | 已生成承诺和范围证明凭证。 |
| 后置条件 | 用于判断值v是否位于范围内 |
| 参数 | V：承诺值，数据类型\*Point。  proof：范围证明凭证，bulletproof结构体。 |
| 返回值 | 验证成功与否。 |