# 中信梧桐港自研分布式账本

# 系统说明

## 1、背景介绍

中信梧桐港供应链有限公司（以下简称“公司”）在可信仓单融资的业务实践中，组织团队自主研发专注于供应链金融、合规、安全、高效、智能的供应链金融分布式账本运行平台，不断探索技术创新应用模式，结合自身项目落地生根，现已投入到实际生产环境中。

相比于已有的fabric hyperledger等技术，公司供应链分布式账本（以下简称“系统”）具有以下特点：

**一是规范，** 始终处于系统设计和实施中第一重要位置。严格遵守《金融分布式账本技术安全规范》（以下简称“规范”）的各项规定（下文有详细说明）。

**二是安全**，采用rust语言语言，在编译阶段就解决了内存泄漏、悬空指针等运行时可能存在的问题。系统采用raft一致性算法，一致得到充分保障。可以适配国密sm2/3/4以及rsa、aes等加密算法，充分考虑传输安全和隐私保护需要。

**三是灵活**，组件可插拔，解耦了区块存储、节点传输、加密、共识、智能合约、RPC远程调用6个微服务，松耦合的架构，便于各组件平滑迁移升，比如新的数据存储或者新的加密方案等。同时也有利于针对一些具体的业务场景，定制特定的功能，适配供应链金融数据监管需要。

**四是高效**，Rust 强调并秉持零开销抽象的理念，在提供诸多高级语言特性的同时，没有引入额外的开销，性能可以媲美 C++。自带端点数据采集，可灵活集成到stm32系列芯片中，实现原始数据一步上链，避免中间环节和人为干预。

**五是便利**，接口更加丰富，提供了多样化的客户端调用，如明文存证、密文存证、签名存证、哈希存证、智能合约调用、以及各种节点控制参数查询合计约40余类接口，充分满足供应链金融多变复杂的业务场景。智能合约开发和区块链部署便利性进一步提高，产品支持一键部署，除了python智能合调用需要的python环境以外，没有其他外部依赖。作为默认智能合约开发语言，python是使用最为广泛、对程序员最友好、最易用的编程语言之一。同时python社区生态极其庞大，借助这一活跃的生态，可以在智能合约实施的同时绑定新的功能。

现根据《金融分布式账本技术安全规范》，在密码箅法、共识机制、点对点通讯协议、分布式存储、基础硬件、苺础软件、密码算法、节点通信、 账本数据、共识协议、智能合约、身份管理、隐私保护、监管支撑、运维要求和治理机制等方面等方面，逐条审核自研产品规范合规性。

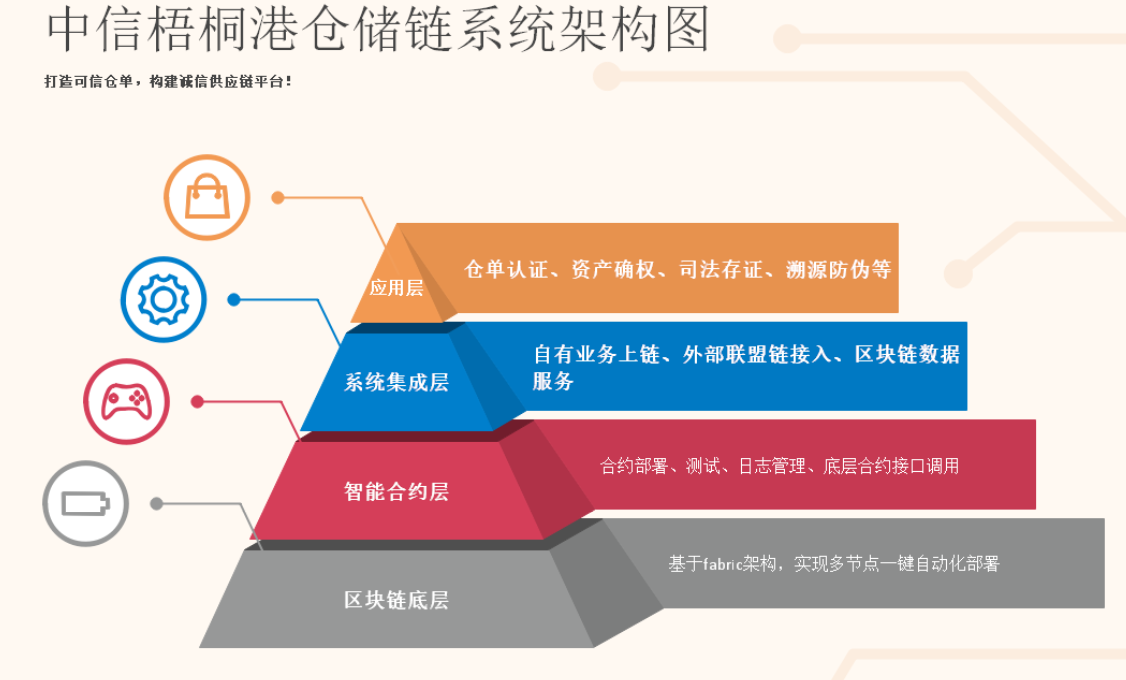
## 2、主要技术架构和使用说明

### 2.1 架构

分布式账本的模块、开发语言和使用说明：

* raft一致性：rust，已实现；
* 传输：rust，已实现，可以选配tls或者基于sm2/4两种加密传输；
* 加密：rust，已实现，基于sm2/3/4，可以定制额外的加密方法；
* 区块存储：rust，已实现，可以适配文本，hadoop，关系型数据库、非关系型数据库等跟踪方式，实现数据同步；
* 远程调用：rust，已实现，可以定制额外接口；
* 合约：rust，已实现，智能合约采用python
* 审计：rust，golang，已实现，使用中可以根据场景定制开发
* CA（证书认证管理，权限管理）：rust，golang，已实现，目前基于sm2，自研证书链生成工具，可以外接硬件加密等。

分布式账本实施应用架构：



### 2.2 配置

系统采用接口调用，接口配置在项目根目录中配置，配置范例：

ORG=org1  
CANAME=ca1  
ROOT=D://myrust/phoenix\_chain  
CAURL=127.0.0.1:9009  
INNER=127.0.0.1:8000  
RPC=127.0.0.1:9000  
FRIENDS=127.0.0.1:8001  
  
CRTDIR=$ROOT/data/crt  
TLSCERT=$ROOT/data/tls/cert.pem  
TLSKEY=$ROOT/data/tls/private.key  
  
CADIR=$ROOT/ca/data/child/$ORG  
CACERT=$CADIR/cert  
CAKEY=$CADIR/sk.pem  
CAPUB=$CADIR/pk.pem  
  
  
SLEDDIR=${ROOT}/data/db  
CONTRACTDIR=${ROOT}/data/contract  
BLOCKDIR=${ROOT}/data/block/  
  
ECNMETHOD=sm  
SAVE=txt  
#INNERTRANS=tls  
INNERTRANS=sm

### 2.3 合约

系统的具体功能实现需要配合智能合约，智能合约书写范例：

import json  
def terms(args):  
 res=args+"\n hello from terms"  
 return res  
  
def after():  
 print ("this is test")  
  
 # must return str no json dumps in return  
 return "ok"  
  
def rollback():  
 return "ok"

terms函数有入参和出参，用于合约测算，在terms函数结果一致时，系统根据调用指令执行after函数操作具体的交易，在失败时执行rollback函数，rollback是after的回滚，需要由客户自己定义

### 2.4 接口

系统提供post调用接口，post参数如下：

#[derive(Serialize, Deserialize, Debug, Clone, Default)]  
pub struct SmartContract {  
 pub contract: String,  
 // url为空表示执行自己  
 #[serde(default = "default\_resource")]  
 pub url: String,  
 #[serde(default = "default\_bool\_false")]  
 pub hash: bool,  
 #[serde(default = "default\_bool\_false")]  
 pub sign: bool,  
 #[serde(default = "default\_bool\_false")]  
 pub after: bool,  
 #[serde(default = "default\_bool\_false")]  
 pub rollback: bool,  
 // 如果是直接存证，数据放到args  
 #[serde(default = "default\_resource")]  
 pub args: String,  
 #[serde(default = "default\_resource")]  
 pub term\_res: String,  
 #[serde(default = "default\_resource")]  
 pub after\_res: String,  
 #[serde(default = "default\_resource")]  
 pub rollback\_res: String,  
 #[serde(default = "default\_vec\_string")]  
 pub signatures: Vec<String>,  
 #[serde(default = "default\_resource")]  
 // 主要是存证后把整个区块放进去，返回给审计模块  
 pub saved\_block: String,  
 #[serde(default = "default\_resource")]  
 pub voucher: String,  
 #[serde(default = "default\_resource")]  
 pub status: String,  
}

python调用代码范例如下：

def smartcontract():  
 contract = dict(  
 contract='test',  
 url="127.0.0.1:8001",  
 sign=True,  
 after=True,  
 args="asdfa",  
 )  
 url = "http://localhost:9000/transaction"  
 a = requests.post(url, json=contract)  
 print (a.text)  
 res1=json.loads(a.text)  
 v=res1["voucher"]  
 time.sleep(4)  
 res2=query\_contract(v)  
 print (res2)  
  
 return res1  
def query\_contract(voucher="76aa9cad-92f4-465d-9495-264197134ada"):  
 url = f"http://localhost:9000/query/contract\_status/{voucher}"  
 res2 = requests.get(url)  
 return res2.text  
smartcontract()  
# query\_contract()

### 2.5 监控

系统自带监控和审计模块，使用flutter编写，可以在ios，安卓，win10，web运行，系统截图如下：

|  |  |
| --- | --- |
|  |  |
|  |  |

## 3、系统建设参考文件

严格参考规范“2 规范性引用文件”部分，包括：

* 信息安全技术网络安全等级保护基本要求
* 信息安全技术5船密码杂凑算法
* 信息安全技术914分组密码算法
* 信息安全技术5似椭阀曲线公钥密码算法
* 信息安全技术个人信息安全规范
* 信息安全技术密码模块安全要求
* 密码应用标识规范
* 似密码算法使用规范
* 密码筅法加密签名消息语法规范
* 基于sm2密码算法的数字证书格式规范
* 密码模块安全技术要求
* 密码模块安全检测要求
* 金融数据密码机技术规范
* 信息系统密码应用基木要求

## 4、**术语定义和本系统对应解释**

* 数字签名：采用SM2算法
* 散列杂凑函数：采用SM3算法
* 对称密码：采用SM4算法
* 非对称密码：采用SM2算法
* 随机数：系统采用的随机数包括UUID4，时间戳、基于均匀分布的随机数等。
* 证书：基于sm2非对称加密，由根证书对子证书认证，从而形成证书链。
* CA证书：专门配备CA模块，可以根据证书链逐级认证。
* 访问控制：基于证书链实现访问控制。
* 零知识证明：sm3哈希加密
* 群签：专门的CA模块，根据业务需求配置证书链，基于sm2。
* 环签名 ：同上。
* 隐私保护：基于sm2、sm3、sm4多种加密存储方式
* 标识密码算法：可以选配sm2或者rsa。
* 同态加密：sm2、sm4、rsa、aes
* 秘密共享：sm2，rsa。
* 对等网络：去中心化
* 共识协议：raft一致性算法
* 分布式账本：可以在多个站点、不同地理位罝或者多个机构组成的网络里实现丼同治理及分享的数据库。
* 分布式账本技术：包括密码算法、共识机制、点对点通讯协议、分布式存储等多种技术。
* 智能合约：python语言编写智能合约，以信息化方式传擂、验证或执行合同，在分布式账本上体现为可自动执行。
* 拒绝服务：可以防止常见拒绝服务攻击。
* 节点：可以选配证书配置、合约加载、合约执行、查询等角色。
* 交易验证节点：基于raft，由“领导”节点负责对提交的交易数据进行验证。
* 共识节点：每个心跳领导节点都会发送数据快照，各个节点根据快照自检和交叉核对，实现数据一致。
* 记账节点：“领导”节点负责账本数据完整。
* 鉴别码：基于随机数、sm2、sm4形成的密码鉴别。
* 数据完整性：选配标准的tls数据传输或者基于sm2和sm4的加密传输，在数据不完整或者篡改的情况下，将无法解析，拒绝服务。
* 保密性：同上
* 不可链接性：签名数据不同保证同一用户在多次交易过程中出示的匿名身份标识，不能够被还原为同一用户。
* 一致性：系统各文档或各部分之间统一的、标准化、无矛盾。
* 终局性：交易一旦确认，就不会被回滚或者撤销。
* 激励相容：一种制度安排，使行为人追求个体利益的行为，正好与整体实现价值最大化的目标相吻合。
* 局部广播：智能合约仅仅发往经过授权的相关方节点，链上存证可以使用hash加密等方式避免信息在整个分布式账本上被泄漏。
* 混淆技术：可以在ca模块建立外部不可见的用户地址id查询，交易对手暴露的仅仅是系统id。
* 局部聚集系数:基于raft协议，在领导节点产生时，所有节点和领导节点交互。
* 原子性：严格记录每步操作用于审计和回滚。智能合约在执行过程中发生错误，会被报告，在智能合约本身实现混滚操作的情况下，可以实现回滚到智能合约开始前的状态。
* 图灵完备：智能合约语言具备图灵完整性。
* 不可否认性：基于sm2的数据签名，区块自带时间戳。
* 标识信息：结合证书链和ca模块，能够单独或者与其他信息组合以识别、追踪到特定自然人身份或反映特定自然人活动情况的信息。
* 隐私信息：可实现隐私保护
* 个人信息：可以以记录特定自然人身份或荇反映特定自然 人活动情况的各种信息。
* 控制者：可分权限层级实现控制
* 干预机制：可以禁止或限制参与金融分布式账本系统的特定角色的特定行为，为异常愔况提供的紧急制动措施。
* 块分组密码：sm4
* 密钥交换：通过随机数和非对称加密在通信节点之间安全地共亨密钥。

## 5、安全体系框架

本系统安全体系包括基础硬件、基础软件、密码算法、节点通信、账本数据、共识协 议、智能合约、身份管理、隐私保护、监管支掙、运维要求和治理机制等方面。在基础硬件、基础软件、密码算法、节点通信、账本数据、共识协议、智能合约、身份管理、隐私保护、监管支持符合规定。具体如下

## 6、基础碩件

### 6.1、硬件设备

可以实现关键节点冗余部署，保证系统可用性。可以接入各类加密硬件厂商提供的加密设备。硬件加密设备应满足如下要求：加密机设备应符合国家密码管理部门颁布的GM/T 0045—2016的要求；个人密码设备（如服灯、加密卡、带证或的移动终端等〉符合行业主管部门和 国家密码管理部门的要求。

### 6.2、网络安全

数据同步基于成熟的分布式数据存储raft一致性算法，共识节点或记账节点之间能直接进行网络通信或能间接进行消息传递;在网络拓扑中，可以防止单个节点故陣而形成网络隔离；保证毎个重要节点具有较大的局部聚集系数。

### 6.3、通信传输安全

可以选择基于tls的https传输或者基于sm2、sm4的加密传输方式，做到：在参与分布式账本的节点之间建立安全传输通道，保证数据传输的完整性和不可篡改性；对数据和信息采取相应的防护措施，保证其能抵抗篡改、重放等主动或被动攻击；采用密码技术保证节点间通信过程中敏感信息字段或整个报文的保密性，确保信息在存储、传 输过程中不被非授权用户读取和篡改；可以采用有权限的网络汸问控制，在参与分布式账本节点之间构述虚拟专用网络，降低网络攻击造成的危害。

## 7、**基础软件**

### 7**.**1、**基本要求**

系统软件环境遵循GB/T 22239—2019中三级以上的主机安全、应用安全、数据安全及备份恢复相 关要求，包括账本结构、共识模块、分布式组网、数据存储、智能合约、接口设计、数据传输、时 间同步和操作系统等方面的要求。

### 7**.**2、**账本结构**

账本结构应具有防篡改性。账本结构宜使用块链式或近似块链式的存储结构，使用哈希嵌套保证数据难以被篡改。账本应具有数据校验功能。任何一条记录被非法篡改后都可通过历史账本数据问溯以快速检验出。

### 7.3、共识横块

基于raft算法，共识模块应能协调各系统参与方有序参与数据打包和共识过程，并保证各参与方的数据一致性。 系统无故故障或欺诈节点时，能在心跳时间内达成一致的、正确的共识，输出正确结果。 在故障节点和欺诈节点的总数量超过现论值的情况下，系统能正常工作。

### 7**.**4、**分布式组网**

系统参与方节点可在物理部署上进行分离，各节点基于网络通信协议和对等网络进行通信和数据互换。各节点独立存储具有一致性的账本数据，且保证任意单个节点故障都不影响整个系统的正常工作。系统由分布在不同地点的节点互连而成，网络中可无中心节点。通信控制功能分布在各节点上，且任一节点均至少与其他两个节点逑立通信连接。

### 7**.**5、**数据存储**

账本数据可以根据数据对象的类别独立存储，账户数据、交易数据、配罝数据以及账本元数据等可以分别存储、分别管理、分别操作。敏感信息可以加密存储，并应有数据访问等权限的控制和管理。节点以证书及其私钥的存储可以私密管理。数据存储可选用结构化数据库、非结构化数据库或混合选用。数据库可以选用各种主流数据库的稳定版本。

### 7**.**6、**智能合约**

智能合约采用python语言编写，可以独立配置可信的软件硬件执行环境。智能合约代码存储和运行时，系统具备相应的安全保护能力，不应允许未授权实体明文读取合约代码和状态。智能合约应具备数据前向兼容的能力，版本迭代时，旧版本的合约应及时停用，并存档数据，新版 本合约应能调用历史数据。智能合约的运行机制宜有前向兼容的能力，当系统版本升级后，智能合约能正常执行。系统通过有效的智能合约审核以确保合约代码所表达的逻辑无漏洞。智能合约的发布可以引入相关方联合审核机制。

### 7.7、**接口设计**

系统设计了良好的接口，隐藏底层账本的细节，为应用层提供简洁的调用方法。接口设计简洁明了，提供完整的功能，能完成交易和维护分布式账本数据，并且有完善的权限管理机制，接口设计应考虑扩展性和兼容性。

### **7.**8、**数据传输**

传输数据过程中，可以使用对称（aes，sm4）或非对称密（sm2，rsa）算法对数据进行加密，防止数据在传输过程中被窃取。可以定制开发其他加密算法。

### 7.9、时间同步

系统保证节点之间的时间戳误差维持在共识协议允许的范围内。可使用经过认证的中心化时间同步源进行节点间的时间同步。

### 7**.**10、**操作系统**

系统主体架构采用rust编写，CA和审计模块采用golang编写，智能合约采用python编写，支持linux各种版本、安卓、windows等操作系统。

## 8、**密码算法**

### 8**.**1**基本要求**

系统中的密码算法主要用于数据安全，即保护数据的保密性、完整性、真实性和不可否 认性，包括SM2/3/4，RSA，AES等各种分组密码算法、流密码算法、非对称密码算法、密钥交换算法、密码杂凑算法和标识密码算法等。分布式账本系统所使用的具体密码算法符合相关国家、行业标准。

### 8**.**2、**保密性**

保密性通过密码加密功能实现，其算法包括对称密码算法和非对称密码算法。通信双方在交换信息时，在建立连接之前，使用密码技术进行会话初始化，通过密钥交换协商会话密钥。在通信过程中，应使用会话密钥对敏感信总或整个报文进行加密，并在加密时采取随 机数填充等技术，避免相同的明文数据在加密后生成相同的密文。在存储敏感的业务数据、身份鉴别数据和密钥数据之前，采用密码技术进行加密。

### 8**.**3、**完整性**

系统中的完整性应用场景包括业务数据和密钥的完整性保护。系统在在对数据处理前检验其完整性，数据完整性通过随机数、非对称加密、哈希加密实现。

### 8.4、**真实性**

使用非对称加密、动态口令或数字签名等方式保障真实性，包括：人员的身份鉴别、节点通讯双方的身份鉴别、网络设备接入时的身份鉴别、登录操作系统和数据库系统的用户身份鉴别、应用系统的用户身份鉴别。

### **8.5、不可否认性**

使用数字签名等密码技术生成可靠的电子签名来保障实体行为的不可否认性，系统中所需的具有不可否认性的行为包括发送、接收、审批、创建、修改、刪除、添加和配罝等操作。不可否认性的应用场录包括：实体行为的确认、背书方对实体行为的背书。

### 8.6、**随机性**

密码算法执行过程使用随机数符合国家相关法规对随机性的要求。

### 8.7**密钥管理**

系统专门配有CA模块，实现了包括对密钥的生成、存储、分发、导入、导出、使用、备份、恢复、归档与销毁等环节的全程管理，符合GM/T 0054—2018中 对密钥管理的要求。ca模块可通过秘密共享算法将密钥分解为多个子密钥分别存储或传输。

## 9、**节点通信**

### 9.1、**基本要求**

分布式账本系统采取节点授权准入的原则，在节点通信过程中保证数据的完整性、保密性。

### **9.2、节点身份验证**

可以符合规范第13章“身份管理”中要求的身份认证机制控制节点的接入。采用密码技术对节点通信双方的身份进行验证。

### **9.3、完整性**

符合国家密码标准的消怂鉴別码算法、数宁签名等密码技术来提供通信中数据的完整性保护和校验。节点间通信协议具备应对通信延时、中断等惝况的处理机制。当检测到数据的完整性遭到破坏时，接收节自动调整区块高度，从领导节点重新获取数据。

### 9.4、通信保密性

在通佶节点建立连接之前，符合国家密码标准的秘钥交换技术来产生双方共亨的工作密钥， 并进行双向身份认证，确保通信节点是信息的真实授权方。通信节点使用密钥对通信过稈中的整个报文或会话进行加密处理。符合国家密码标准的技术来违立安全通信通道，避免因传输协议受到攻击而出现的保密性破坏。

## 10、账本数据

### 10.1、完整性

保证账本数据的生成、传输、存储、调用等操作不可被非授权方式更改或破坏。

### 10.2、—致性

分布式账本中记账节点的账本数据保持一致。对账本数据的写入和修改，须经各节点达成共识，以确保各节点的数据一致性。当出现数据分叉时，存在可用规则进行数据选择。

### 10.3、保密性

釆用密码技术保证账本数据中的敏感数据在传输和存储过程中的保密性。账本数据中敏感数据的保护密钥和账本数据本身分开保存，并且保护密钥应支持存放在安全的密码模块中。

### 10.4、有效性

各记账节点的账本数据应符合规范第11章“共识协议”中要求的共识协议保证账本数据的有效性，且满足以下有效性要求：可以对节点存储的账本数据的有效性进行校验；当某个节点的账本数据失效时，可以使用符合规范第11章“共识协议”中要求共识协议保证账本数据的有效性。

### 10.5、账本数据冗余

系统采用去中心化分布式存储，保证账本数据在系统中具有冗余性，防止因竽个节点失效而造成总账本数据的丢失。

### 10.6、访问与使用

系统确保账本数据不被未授权的第三方获取，数据访问和操作符合规范第14章“隐私保 护”中对认证授权，访问控制等方面的技术要求。

### 10.7、安全审计

记账节点对账本数据的橾作满足以下安全审计要求：账本数据的访问提供安全审计功能，审计记录包括访问的日期、时间、用户标识、数据内容等审计相关信息；数据变更提供审计功能，审计记录不仅包括数据变更成功的记录，还包括数据变更失畋的记录；节点有效性校验失畋、一致性校验失畋等情况下同步账本数据，提供安全审计功能，审计记 录包括事件类型、原因、账本数据同步的节点、账本数据校验值等相关信息；审计记录可由记账节点自行记录，不必写入账本。

## **11、共识协议**

### **11.1、基本要求**

可以根据业务特点选用适宜的共识协议，包括但不限于工作量证明、权益证明、授权股权证明、拜占庭容错等，满足不同共识协议安全运行所必需的前提要求。

### **11.2、合法性**

确保参与共识过程的节点经过验证，保证节点共识过程的加入和退出的合法性，以及节点ID与节点实体的一一对应，形成可信节点。

### **11.3、正确性**

底层数据一致性采用raft，已经经过公开安全评估，协议算法的测试全面完整，可信节点应为协议算法的运行提供安全可信的硬件软件（如服务器、操作系统等）基础，确保协议 算法运行环境的安全性与可靠性。

### **11.4、终局性**

算法在可接受的有限时间内具有终局性，如在一定时间无法选出领导节点，就会开始下一轮选举。所有参与共识的可信节点，经过一段可接受时间内的交互，最终达成一致性结果。

### **11.5、一致性**

所有参与共识的可信节点得到的结果是相同的，符合共识协议。

### **11.6、不可伪造性**

系统中恶意节点占比不超过共识协议容错率时，任何对系统当前状态进行恶意构造以欺骗其他可信节点所需要的时间，不少于可接受范围。

### 11**.**7、**可用性**

协议应具备抗DDoS攻击、处理恶意报文、识别恶意节点的能力，可以采取不转发、拒绝连接、黑名单等措施缩小影响，使系统获得一定的主动防御能力，提高系统的可用性。系统能始终在正常时间内对客户端的请求进行响应。

### 11**.**8、**健壮性**

数据在恶意攻击后被污染时，被攻击节点成通过与系统中其他可信节点交互等方式来检测出攻 击及数据污染。系统中的节点如遇到网络故障等情况与系统断幵连接，可能会出现与系统中其他节点状态不一致的 情况。在恢复连接后，通过与系统屮其他可信节点交互等干预方法，保证节点数据恢复正常状态且受攻击前的数据不会丢失，并保持与正常节点间数据的一致性。

### 11**.**9、**容错性**

系统屮恶意节点占比不超过共识协议的容错率时，系统保证正常运作，且保持数据一致性。

### 11.11、可监管性

单次共识过程和系统运行的整个共识历史都成可审计、可监管，该历史应不可被驀改。

### 11.11、低延迟

共识协议保持低响应延迟，满足金融系统对于数据同步的时间要求。

### 11**.**12、**激励相容**

采用激励机制保障系统安全，计算系统可承栽的价值上限，并对其上的应用进行检査，避免超过安全阈值。

### 11**.**13、**可拓展性**

协议具备动态拓展能力，允许在系统保持正常服务的前提下动态或静态增删节点。

## 12、**智能合约**

### 12.1、**基本要求**

支持图灵完备智能合约和非图灵完备智能合约，两者都符合本章安全要求。

### 12.2、**版本控制**

在源代码屮通过金融分布式账本指定的方式定义版本号。在配罝文件中定义版本号，该配罝文件应与智能合约代码一同部署。在部署或升级操作时定义版本号。智能合约升级后，在金融分布式账本中保留前版本。交易信息中明确调用的智能合约版本。

### 12.3、**访问控制**

有相应的机制控制用户对智能合约的访问。有相应机制在支持智能合约之间相互访问的条件下，限制错误智能合约的感染。有相应机制控制智能合约对外部环境的访问。对智能合约提供隔离的执行环境。

### 12.4、**复杂度限制**

可以从合约源代码总长度、资源消耗和执行时间等方面限制合约代码的复杂度。

### 12.5、原子性

智能合约的执行有原子性，支持执行过程中发生错误时的回滚操作。一旦出现异常，所有的执行可以回撤，以避免中间态导致数据不一致。

### 12.6、 一致性

智能合约执行具备一致性，合约在所有金融分布式账本网络节点上的执行结果完全相同。 多个竹点同时实现合约时，保证数据的完整性且数据同步不相互干扰。

### 12.7、**安全审计**

智能合约的安全审计和评估对象包括智能合约设计与业务逻辑安全、源代码安全审计、编译环境 审计及相关的应急响应机制等。智能合约经过相关专业技术人员的审计，并保留审计记录。

### 12.8、**生命周期管理**

从部署到废止的生命周期满足以下要求：有相应机制控制智能合约的部署行为，防止恶意部署智能合约；提供智能合约的冻结功能，防止智能合约的漏洞持续影响系统；提供智能合约升级方案和机制以修复智能合约的漏洞；提供智能合约的废止功能；支持权限可控的智能合约升级方法；支持从金融分布式账本中获取与合约相关的原始数据来解析智能合约在金融分布式账本上的业务数据；在合约更新升级、重新部署后，能安全地将原合约数据迁移至新合约。

### 12.9、**攻击防范**

有相应机制保证系统能对抗由智能合约引起的DDoS攻击，防止其长时间占用资源。有相应机制保降在系统遭受攻击、服务受到影晌时，智能合约的运行可被干预（比如运行计时）。有相应机制防止隔离执行环境中的智能合约访问其执行环境之外的资源，比如设置python运行环境。

### 12.10、**安全验证**

可以基于智能合约安全规则库和问题合约模式库实现智能合约的漏洞检测，实现基于安全规则和配罝倍息自动生成安全智能合约模板的机制，通过形式化方法验证智能合约代码的正确性。

## 13、**身份管理**

### 13.1、基本要求

可以通过在上层搭建应用，实现有效的用户身份管理，主要功能包括身份注册、身份核实、账户管理、凭证生命周期管理、 身份鉴别、节点标识管理和撤销等。同时，保证身份信息的安全性，并对身份进行监管审计。

### 13.2、身份定义

支持身份数宇化标识。支持普通成员账户、系统管理员账户和其他特定权限的系统用户账户，其中系统管理贝账户具有最髙权限（如部署智能合约，一个身份可对应多个账户。每个账户关联一个身份标识。

### 13.3、身份注册

通过继续搭建上层应用，支持规范13.3中身份注册相关要求。

### 13.4、身份核实

通过继续搭建上层应用，支持规范13.4中身份核实相关要求。

### 13.5、账户管理

通过继续搭建上层应用，支持规范13.5中账户管理相关要求。

### 13.6、凭证生命周期管理

通过外接加密设备，搭建上层应用，支持规范13.6中凭证管理相关要求。系统CA模块可以基于SM2或者openssl，自生成证书链，实现证书签名、校验、认证、非对称加密解密等基本功能。

### 13.7、身份鉴别

通过外接加密设备，搭建上层应用，支持规范13.7中身份管理相关要求。系统CA模块可以基于SM2或者openssl，自生成证书链，实现证书签名、校验、认证、非对称加密解密等基本功能。

### 13.8、节点标识管理

系统支持使用uuid、url、公钥等方式作为通信节点唯一的节点标识，支持指定节点角色，支持基于证书链的节点许可，支持节点之间建立数据通信连接之前，先通过标识鉴别信息实现双向身份认证，并建立-条安全的数 据通信信道，该过程应符合“密码算法中对保密性和完整性的要求，节点标识认证失败时结束通信。

### 13.9、身份更新和撤销

提供实体对身份信息进行更新和撤销的功能。当实体对身份信息进行更新时，可以实现针对身份信息进行重新核实。

### 13.10、身份信息安全性

支持将身份数据元素存储在不同的系统中，并在相应系统中受到不同等级的保护，保护要求应依据具体业务进行特定说明。支持根据金融业务需求制定身份数据保密性要求，确保数据不暴露给未经授权方。提供基于属性的访问控制，在数据对象的整个生命周期中身份数据始终保持保密性、完整性， 和可验证性。

系统自带证书链工具具备授权功能作为访问控制的扩展。具有管理密钥创建、派生、分发、存储、安全性和其他管理安全审计功能。身份密钥的管理应遵循金融行业密钥管理标准。用户身份密钥采用符合金融市场所需的目标级安全要求的加密算法及密钥长度。交易具备时间戳，保证正确的交易顺序。

支持对身份、账户、凭证的访问和更改提供安全审计功能，审计记录包括访问的时间、用户标识、数据等审计相关信息。

## 14、隐私保护

### 14.1-3、信息加密验证

系统支持规范14.1规定的隐私保护功能实现。可以通过证书链对交易方身份信息进行标识和鉴别，确保交易方无法被冒用，通过签名算法或者hash算法确保任何人能进行有效性和正确性的验证。系统节点信息传输全部加密，支持除了参与方以及审计方拥有对加密信息解密验证的能力，他人无法从加密信息获取任何其他信息，能够确保除交易参与方外他人无法伪造加密佶息。，确保任何人可对加密的信息进行有效性和正确性的验证。

### 14.4、隐私保护技术要求

支持隐私保护技术要求，包括但是不限于：

* 信息采集时有提示信息，并明确告知客户哪些个人信息会被采集。
* 信息采集时包含客户勾选同意或确认的操作步骤，有明确授权。
* 信息采集时默认对身份标识信总进行部分隐藏，同时提供全部显示手段。
* 信息采集时对客户和采集的佶息进行匹配认证，并对完整性进行校验。
* 信息采集时明确告知收集信息的处理方式、存储期限、锊能合约逻辑内容。
* 信息传输时对信息进行全景加密，加密的密钥和证书不能采用信息传输的同一传输通路进行传递；
* 停止运营产品或服务时，及时停止收集数据的活动，并及时告知客户和为客户提供信息注销 不可见的手段，并向其他节点或组织发布停止运营和处罝数据的信息；
* 密钥发送客户后明确告知其妥菩保钤密钥，并提供密钥更换手段；
* 信息存储时对客户的隐私信息进行加密；
* 信息存储时对客户身份标识信息进行桷要存储；
* 信息在第三方存储时告知客户并获得客户授权；
* 信息展示时对客户身份标识佶息进行隐藏，可额外提供全显示手段，非密文展示釆取 去标识化措施。
* 对非本人展示先获得信息所有者的授权，并对展示人进行认证。
* 信息使用时，明确记录使用者、使用内容、使用时间、使用频率。
* 信息向外部扩散时，告知客户并获得授权，并提供给客户延缓甚至中断扩散、减少扩散影响 的手段。
* 向客户提供信息备份和导出的手段，备份和泞出的信息加密处理,并向客户提供解密手段。 向客户提供信息注销不可见的手段。
* 信息注销不可见时敁获得客户认证和授权。
* 信息加工后产生的信息，也满足上述各项要求。

### 14**.**5、**隐私保护监控与审计**

系统可以定制审计模块，一切操作留有痕迹。支持规范14.5关于隐私保护监控与审计的各项功能实现。

## 15、**监管支撑**

### 15**.**1、**基本要求**

系统具有架构去中心、数据多副本、交易点对点、记录不可篡改的特点。

### 15**.**2、**系统监管**

支持监管机构的接入，以满足信息审计和披齑的要求。支持监管部门的监管活动，包括但不限于设罝监管规则，提取交易记录，按需査询、分析特定业务数据等。支持监管机构访问最底层数据，实现穿透式查询。

### 15**.**3、**信息管理**

支持还取匿名标识中的用户真实身份以及相关交易信息。

### **15.4、事件处理**

当系统或交易出现问题时，能主动报警信息。

### 15**.**5、**交易干预**

具备限制交易权限、冻结账广等功能，为监管机构提供交易干预的技术手段。

### 15**.**6、**智能合约监管**

能按需将监管要求编码写入智能合约强制执行。能根据需要为监管机构提供交易行为统计数据，评价智能合约所提供服务的合规性。

## 16、**运维要求**

支持对节点运行状态以及节点与其他节点的连接进行监控。支持对节点进行高可用的架构，用来应对部分节点的异常。如发现部分节点运行异常，在不影响服务运行的同时及时进行问题的排查，在规定的时限内将节点恢复至正常状态，以保证业务的正常运行。支持收集系统中运行的状态数据，包括节点的远端同步节点数、账本同步平均耗时、节点的健康状态 等，并写入日志供测试或者审查用，如异常应及时预聱并处理。支持发现恶意或者欺诈节点传播恶意损坏的账本数据，或者有节点篡改账本，先定位节点位罝，获获取详细信息，上报委员会。

## 17、**治理机制**

配合适当的管理流程，系统支持规范17章关于治理机制的各项要求。