Source Blockchain
根 源 链

根源链·技术白皮书

基于数据信息追溯及确权的泛物联网应用公有链基础设施

版本 v1.1.08

修订日期 2018.9.8

根源链团队

目录

0 通用术语表

术语/缩略语	英文释义	中文释义
SC	Source Chain	根源链
BOS	Blockchain Operating System	区块链操作系统
SCI	Source Chain Infrastructure	根源链基础设施
PoW	Proof of Work	工作量证明
PBC	Public Block Chain	公有链
LLS	Large Legacy System	大型设施系统
PDL	Protocol Description Language	协议描述语言
IDL	Interface Description Language	接口描述语言
Off-Chain	Out of Block Chain	链下，区块链之外
On-Chain	On the Block Chain	链上，区块链之上
WD	Wallet Driver	钱包驱动器
GW API	Gate Way Application Programming Interface	网关应用程序接口
SDK	Software Development Kit	软件开发包
ASL	Application Service Layer	应用服务层
BSL	Baseline Service Layer	基础服务层
HDL	Hardware Driver Layer	硬件驱动层

SCA	Side Chain Adapter	侧链适配器
2WPP	2 Way Peg Protocol	双向锚定协议
MI	Management Interface	管理接口
AI	Account Interface	账户接口
Hash	Hash	散列，哈希
GPU	Graphics Processing Unit	图形处理单元
OpenCL	Open Computing Language	开放式计算语言
CUDA	Compute Unified Device Architecture	统一计算设施架构
FPGA	Field Programmable Gate Array	现场可编程逻辑门阵列
SCT	Smart Contract Template	智能合约模板
CP	Cryptographic Protocol	加密协议
DCP	Distributed Consensus Protocol	分布式共识协议
MMPP	Miner and Miner Pool Protocol	矿工矿池协议
SCPL	Side Chain Protocol Layer	侧链协议层
SCIL	Side Chain Implementation Layer	侧链实现层
PL	Protocol Layer	协议层
BCL	Block Chain Layer	区块链层
BLL	Business Logic Layer	业务层
CL	Cryptographic Layer	加密层

Compiler	Compiler	编译器
Interpreter	Interpreter	解释器
Key	Key	密钥
Account	Account	账户
Config	Configuration	配置
Monitor	Monitor	监控
DashBoard	DashBoard	仪表盘
SDKPL	SDK Protocol Layer	SDK 协议层
SDKIL	SDK Implementation Layer	SDK 实现层
ASIC	Application Specific Integrated Circuit	专用集成电路
CPU	Central Processing Unit	中央处理单元
ABI	Application Binary Interface	应用二进制接口
Arch	Architecture	架构
x86	x86	x86 硬件架构
MIPS	Microprocessor without Interlocked Pipeline Stages	MIPS 硬件架构
ARM	Acorn RISC Machine	ARM 硬件架构
DES	Data Encryption Standard	数据加密标准
3DES	Triple DES	3 密钥 3 次加密
AES	Advanced Encryption Standard	高级加密标准

RSA	Rivest-Shamir-Adleman	RSA 加密方法
DSA	Digital Signature Algorithm	数字签名算法
ECC	Elliptic Curves Cryptography	椭圆曲线加密学
MD5	Message Digest algorithm 5	单项散列算法 5
SHA	Secure Hash Algorithm	安全哈希算法
scrypt	scrypt	典型：莱特币
scrypt-c	scrypt-cacha	典型：ya 币
ETHASH	Ethereum hash	典型：以太坊
ECDSA	Elliptic Curve Digital Signature Algorithm	椭圆曲线数字签名算法 典型：瑞波币
X11	XCurrency 11	典型：达世币
SHA-256	SHA 256	典型：比特币
lyraz2	lyraz2	典型：零币
ECDH	Elliptic Curve Diffie–Hellman key Exchange	椭圆曲线 D-H 密钥交换
SM1	Secret Management 1	国密 1
SM2	Secret Management 2	国密 2
SM3	Secret Management 3	国密 3
SM4	Secret Management 4	国密 4

HMAC	Hash-based Message Authentication Mode	哈希运算消息认证码
AES-CRT	AES Counter	AES 计算器模式
AES-ECB	AES Electronic Codebook Book	AES 电码本模式
AES-CBC	AES Cipher Block Chaining	AES 分组链模式
AES-CFB	AES Cipher FeedBack	AES 密码反馈模式
AES-OFB	AES Output FeedBack	AES 输出反馈模式
DDOS	Distributed Denial of Service	分布式服务拒绝攻击
Stratum	Stratum	Stratum 矿池协议
DAO	Distributed Autonomous Organization	分布式自治组织
DApp	Distributed Application	分布式应用
SCLSR	Source Chain Link State Routing protocol	根源链链路状态路由协议
SCFD	Source Chain Forward Daemon	根源链转发守护进程
SCFace	Source Chain Face	根源链分布式转发接口
IT	Information Table	根源链分布式转发信息存储表
WST	Waiting Sender Table	根源链分布式转发等待发送列表
FIT	Forward Information Table	根源链分布式转发信息询路表

PGP	Pretty Good Privacy	PGP 加密协议
RSK	Root Stock	RSK 合约侧链实现
MAST	Merklized Abstract Syntax Tree	梅克尔抽象语法分析树
CP	Counter Party	对手方合约方法
LN	Lighting Network	闪电网络合约方法
SPV	Simplified Payment Verification	简单支付验证
RSMC	Revocable Sequence Maturity Contract	到期可撤销合约
HTLC	Hashed Timelock Contract	哈希时间锁定合约
SCNC KMS	Source Chain NC Key Management System	根源链分布式密钥管理系统

1 根源链

根源链是一个基于数据信息追溯及确权的泛物联网应用公有链基础设施。

根源链全称为根源链 BOS（Source Chain Blockchain Operating System），一般简称为根源链。根源链使用区块链技术来实现对社会经济协作与泛金融活动过程产生的信息和数据进行溯源、登记确权以及实现数据交易，是一个社群性公有链，具有安全可靠、稳定性高、可拓展性强的技术特点。

2 概述

2.1 背景

随着信息化时代的发展，各类社会活动均会产生大量的信息和数据，如金融服务、大宗交易、商品贸易、日常消费、产品质量与供应链管理、社交互动、电子政务、游戏活动以及物联网/互联网安全等,这些信息数据往往经过了各种各样的“包装”变成了一种可以交易的资产被社会广泛地接受.数据变成资产后,具备了流动性,金融属性就变得很强, 围绕数据进行交易成为了催生信息产业升级的元力, 也成为了滋生信息的暗网交易、个人隐私数据泄漏、基于大数据杀熟营销、行业溯源造假等一些灰色的产业生态.

另外，由于行业竞争力加剧,社会信用机能不足、企业保护商业隐私等原因使得业务发展过程产生的数据和信息无法联通、不能共享与交换,成为了制约大数据产业应用的障碍。如何明确数据信息主体、客体的权责边界;如何形成良好的数据开发和使用氛围，处理好安全和发展的关系。成为了根源链建设公有链的初衷及未来践行的方向。

2.2 应用生态

根源链坚持聚焦在产业应用主航道,将为不同领域的客户提供通用的分布式数据接入组件, 并逐步与产业应用生态伙伴研发诸如产品溯源、数据确权、分布式电商网络、轻量应用挂载平台等落地应用。

根源链社区在建设应用生态时，主张开放、合作、共赢, 与产业伙伴合作创新、扩大产业价值。在大数据确权交易、食品药品溯源与流通、企业供应链管理、文化资产版权管理、电子政务、智慧城市建设、文娱游戏开发、广告营销、供应链金融、商业积分、快销品 B2B 销售等领域，会逐步形成完善的行业解决方案,并将根据技术与应用特点，不断扩展应用范围，持续进行底层技术与生态的升级。

3 技术要点

根源链 BOS 属于典型的公有链基础设施范畴。参考 wikipedia 给出的描述，公有链定义如下：

Public blockchains：

A public blockchain has absolutely no access restrictions. Anyone with an internet connection can send transactions to it as well as become a validator (i.e., participate in the execution of a consensus protocol). Usually, such networks offer economic incentives for those who secure them and utilize some type of a Proof of Stake or Proof of Work algorithm.

Some of the largest, most known public blockchains are Bitcoin and Ethereum.

因此，作为公有链，其技术要点可简单归纳为：

(1) 不限制参与者，更不限制时空位置

(2) 权益证明算法，例如 POW

(3) 经济激励

根源链 BOS 作为基础设施符合人类社会在数字空间的基础，其特性简单归结为：

(1) 新型分布式计算范式

(2) 匿名不可篡改

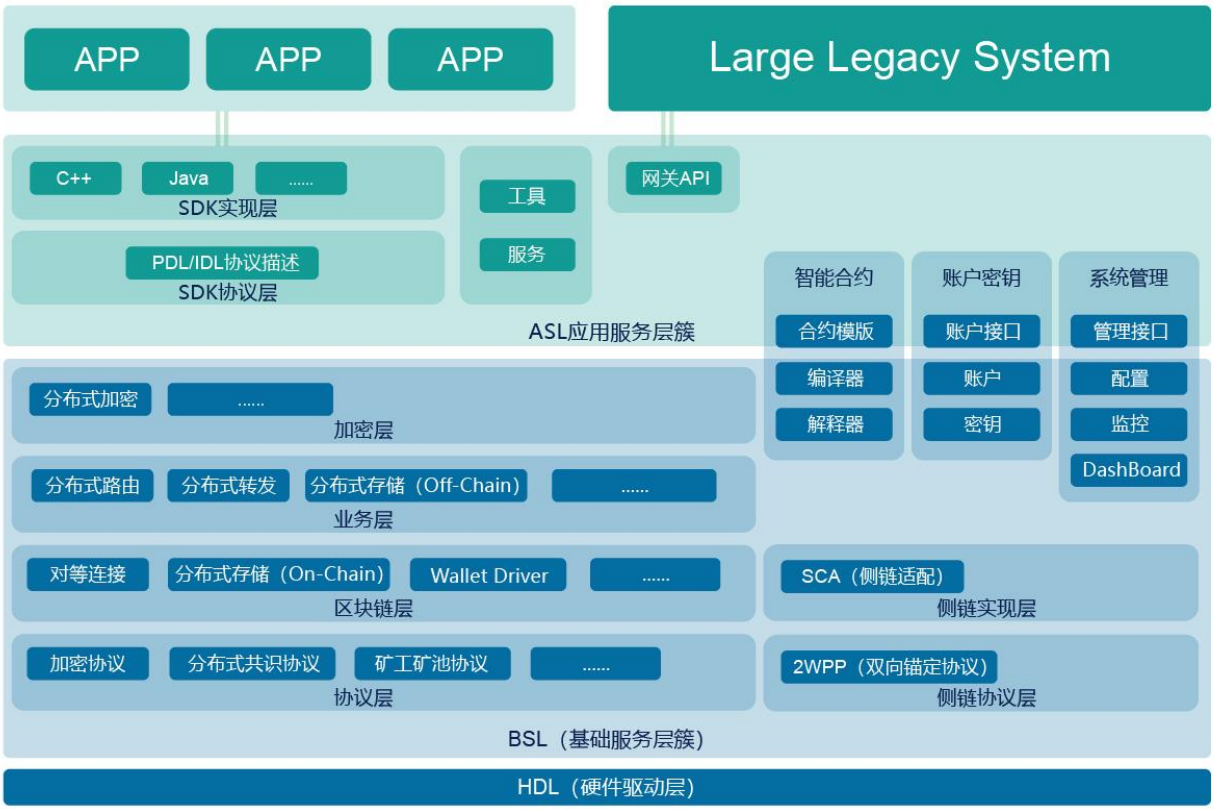
(3) 组件可插拔设计

(4) 新型溯源规范

(5) 激励策略池

而根源链则基于通用型区块链基础设施，结合物联网及应用化需求进行了必要的技术优化与扩充。以下对根源链的系统架构及其各部分技术要点进行简要解释。

3.1 系统架构



3.2 硬件驱动层 HDL

区块链计算机硬件驱动是硬件加速引擎激活的前提，定制化的区块链计算机具有深度挖掘软硬件的性能潜力，大幅提高区块链系统性能，表现为：

(1) Hash 计算

(2) 硬件计算：GPU 编程(openCL、CUDA)、FPGA 编程(Field - Programmable Gate Array)、ASIC 编程(Application Specific Integrated Circuit)

(3) CPU 计算：X86 体系、MIPS 体系、ARM 体系

(4) 加解密计算：使用专用加解密板卡提高加解密性能

3.3 协议层

协议层是根源链 BOS 的核心，不仅具备抽象可插拔的能力，而且灵活、低耦、易扩展。

3.3.1 加密协议

内容包括：

(1) 签名验证算法（RSA,ECDSA,SM2）

(2) 数字信封算法（RSA,ECDH,SM2，NULL）

(3) 哈希、MAC 算法（SHA256，HMAC,SM3）

(4) 对称加密算法（AES-CBC,AES-ECB,SM4，NULL）

其中括号内为目前支持的算法，包括美标和国密两种标准，更多的算法还在继续扩展中。

3.3.2 分布式共识协议

为了易于扩展分布式共识算法，根源链 BOS 提出了分布式共识协议，从而便于任何共识算法的接入。换言之，分布式共识协议促使任何共识、一致性算法作为插件接入到根源链 BOS 中。

3.3.3 矿工矿池协议

目前矿工支持包括：Mining software，BFGMiner，CGMiner，libblkmaker

目前矿池支持包括：ckpool，Eloipool，Stratum-Mining

3.4 区块链层

区块链层承载的是根源链公链的核心，包括对等连接、分布式存储(On-Chain)、钱包驱动(Wallet Driver)。

3.4.1 对等连接

根源链 BOS 的对等连接主要体现在去中心化的底层所使用的网络拓扑，实际为 P2P 的网络，正常时 P2P 网络是很难被 DDOS 攻击的，网络中的所有节点都是相同角色，并不存在某个有着特殊含义的中心化节点，所以攻击者找不到特定的攻击目标来实现 DDOS 攻击。

从防护安全的角度出发，对等连接和 DDOS 攻击是根源链 BOS 重点考虑的。

(1) 攻击只会影响到单个节点。

单个节点很可能会因为 DDOS 攻击而导致服务不可用，单个节点的下线对整个网络而言，影响可以不计。只要该节点恢复，马上就会自动连接到 P2P 网络，并且从网络中同步因为 DDOS 攻击而没有同步的区块数据，相应的服务能很快恢复。

(2) 攻击只能针对于网络通讯底层

如果攻击者采用区块链网络协议对节点进行 DDOS 攻击，寄希望于通过协议使得 DDOS 的影响能自动洪泛于 P2P 网络，但这种是不可行的。

- Case A: 如果攻击者构造异常的数据包，那么在该数据包到达的第一个节点，节点就会对数据包进行校验处理，校验不通过将直接丢弃报文，使得攻击的效果仅限于网络中的第一个节点。
- Case B: 如果攻击者伪造为 P2P 网络中的一个节点，向 P2P 网络发送异常的数据包，和 A 类似，该报文在和伪造节点相连的一个或多个节点被抛弃，而且此时相邻节点在验证出错误报文后，会对报文的来源节点启用惩罚机制，当达到惩

罚的阀值之后，自动关闭和该节点的连接。这样很快的时间内，伪造节点就会被从 P2P 网络中剔除，整个 P2P 网络不会有任何影响。

(3) 正常攻击报文的成本因素

如果攻击者构造正常的交易数据包和区块数据，交易中不管是查询类型还是创建类型，都需要耗费相应价值的数字货币，这种攻击的费用代价相当高，而费用会自动转移到被攻击的钱包，最后的效果是攻击者没有达到攻击的效果，被攻击的一方反而赚的盆满钵满。

3.4.2 中继激励

角色定义：

要素名	含义
Client	客户端，属于使用匿名服务的用户，细分为资源的提供方（Provider Client），资源的消费方（Consumer Client）
Relay	中继端，属于提供匿名中继的服务侧，细分为 Entry，Middle，Exit 三类。 Entry 的含义是靠近 Provider Client 的中继 Middle 的含义是具备邻居的中继 Exit 的含义是靠近 Consumer Client 的中继
Assignment Server	分配服务器，属于提供建立 Consensus Group，以及间接建立 circuit 的服务主体

Consensus Group	共识组，属于分配服务器提供 SCPATH 能力的实例，1）只要连接到分配服务器的客户端和中继，都能够提供它们的公钥到共识组 2）执行这些公钥的验证 3）输出一个 circuit，并将该 circuit 给到 circuit 中定义了的客户端们。 任何共识组的生成都是临时的。
circuit	环路/线路，包含了客户端和若干中继的数据结构。分配服务器需要依赖这个 circuit 作为输入，输出最终的路径搜索结果（Path Lookup）。 任何 circuit 的生成都是临时的。

安全性评价：

要点名	含义
Anonymity	匿名性
Group Formation	（共识）组的编排方法
Circuit Diversity	环路/线路的多样性
Persistent Guards	持久防护性

匿名中继激励要点：

1）只有在 circuit 上的客户端和中继才被允许挖矿。

2）挖到的币具有特殊的分配规则：基于 circuit 上所有的客户端和中继，按照各自吞吐率分配所获币。

匿名中继激励过程：

1）每个 Client 和每个 Relay 生成自己的临时密钥 R，并计算该密钥的 Hash

2）客户端发送一个元组（coin#，Rc）作为数据包，发送到这条 circuit，而且严格限制数据包数量必须达到 m

3）在这条 circuit 中，每个中继都向元组发送自己的密钥的 Hash 值，分别表示为 Re，Rm，Rx，并且每个中继都将该元组发送到下一个中继

4）在这条 circuit 中，Exit Relay 生成一个币提交数据包 B=(coin#，Rc，Re，Rm，Rx)

5）在这条 circuit 中，Exit Relay 使用自己的临时公钥去签这个 B 数据包，表示为 Sxb。并且发送元组（B，Sxb，Rx）到 Middle Relay。

6）在这条 circuit 中，每个参与者都区使用自己的公钥签数据包 B，表示为 Sib，并将这个数据注入到元组中，最后再将这个元组转发到当前参与者的前驱参与者。

7）在这条 circuit 中，客户端生成 PoB，表示为 P=(B，Sxb，Smb，Seb，Scb，Rx，Rm，Re，Rc)，这说明任何参与者都可以验证 Client，Entry，Middle，Exit 四个参与者的公钥。

8）在这条 circuit 中，客户端进行 Hash(CSi，B，Rx，Re，Rm，Rc)的值的低位区数值是否等于 0，表示 BSTK 已经被挖到。

并验证 BSTK 的有效性。

9）最终，在这条 circuit 中，客户端依旧使用根源链的方式，以 BSTK 支付这条 circuit 上的每个中继。

3.4.3 分布式存储（On-C）

分布式存储(On-Chain，链上)有别于分布式存储(Off-Chain，链下)，因为链上数据是稀有的，所以链上数据具备特殊性。根源链 BOS 针对链上数据进行了独特的设计，主要依赖的技术包括 OP_RETURN 和 MultiSig。

根源链 BOS 使用 OP_RETURN 作为链上数据的选择主要是出于两点需要。第一是 BSTK 销毁，第二是 PoE 存在证明。根源链 BOS 扩展了 PoE 的能力，使其具备了会计属性(记账、查账、对账、审账、分账和销账)。

根源链 BOS 使用 MultiSig 作为链上存储是源于 P2SH 的衍生产物，对 n-of-m 型的多签名地址，因为只需要提供 n 个有效签名就可以花费该交易，但剩下的 m 减 n 个签名的内容还是分配了存储空间，根源链 BOS 巧妙地设计并重新利用这些剩余的存储空间，这种方法的好处是该交易依然可以继续被花费。

3.4.4 钱包驱动

根源链 BOS 重新构建了钱包使其具备驱动能力，建立密钥与账户的映射关系。钱包驱动的好处在于面向数据所有权方面提供了密钥、账户所需要的接口，而面向用户层面使用所有权时并不关心驱动是如何调度的。

我们知道权益是通过数字密钥、地址和数字签名来确立的。密钥的独立特性使得密钥数据库也具备独立特性，钱包的本质就是密钥数据库(KeyStore)。根源链 BOS 通过重建密钥数据库，使得上层模块调度请求时，钱包驱动能够灵活地去读写账户密钥模块。

3.5 业务层

业务层是根源链 BOS 的功能核心，不仅引入了许多先进技术，而且自主实现了多领域、多场景、易定制的诸多模块。主要包括分布式路由、分布式转发、分布式存储(Off-Chain)，以及业务层接入的核心术语单元。

(1) Blocklabel，作为根源链 BOS 区块网络命名规则（SCBN，根源链区块网络）的一部分，用于标识根源链 BOS 全局、唯一的一般数据或资源；它的形式表示为 URI，以/为分隔符，分隔符之间是元素；例如，/sourcechain/bstk/what/do/you/want/to/do；这种形式不仅简单易理解，而且容易扩展，最关键的是它作为根源链 BOS 核心协议词之一；此外，命名所带来的优势是传统 P2P 网络无法比拟的；

(2) Market，用于表示市场的参与者角色；其中，参与者可以是纯粹的生产者或者消费者，亦可以是产销共同体；

(3) Contract，用于验证交易的数据类型，由合约代码组成，合约代码包含数据、行为、输入和输出；具体定义参考合约模板章节；

(4) Price，用于 Blocklabel 对应数据的定价，在 Market 的参与下，促进信息披露与价值流通；Price 的引入与分布式存储(Off-Chain)有关，用于标识 BlockLabel 指向的资源的定价；

使用命名技术和区块网络的目的是友好的分布式实现、安全易扩展及数据标识等，从而形成一个具备区块链属性的 Label-NDN 分布式存储网络。

分布式实现不同于 P2P 并不意味者两者之间的优劣。业务层引入命名化的分布式计算是为了有效驱动分布式存储(Off-Chain)的读写与数据传输，而面向应用层面是透明的。

安全易扩展是区块网络的重要特性之一，原因在于基于 IP 网络安全需要对终端和连接同时信任。IP 网络接受任何人发送的任何内容，不管数据包的内容，只要发送者看似合法，这种情况导致恶意信息发送到接收者，这是 IP 网站容易被攻击的根源；通过签名

加密了关于数据请求者的信息，除非点对点链路直接连接到发出请求的主机，否则路由器将只知道有人请求某些数据，但不知道是谁发起请求，具备一定的匿名性(Anonymous)。

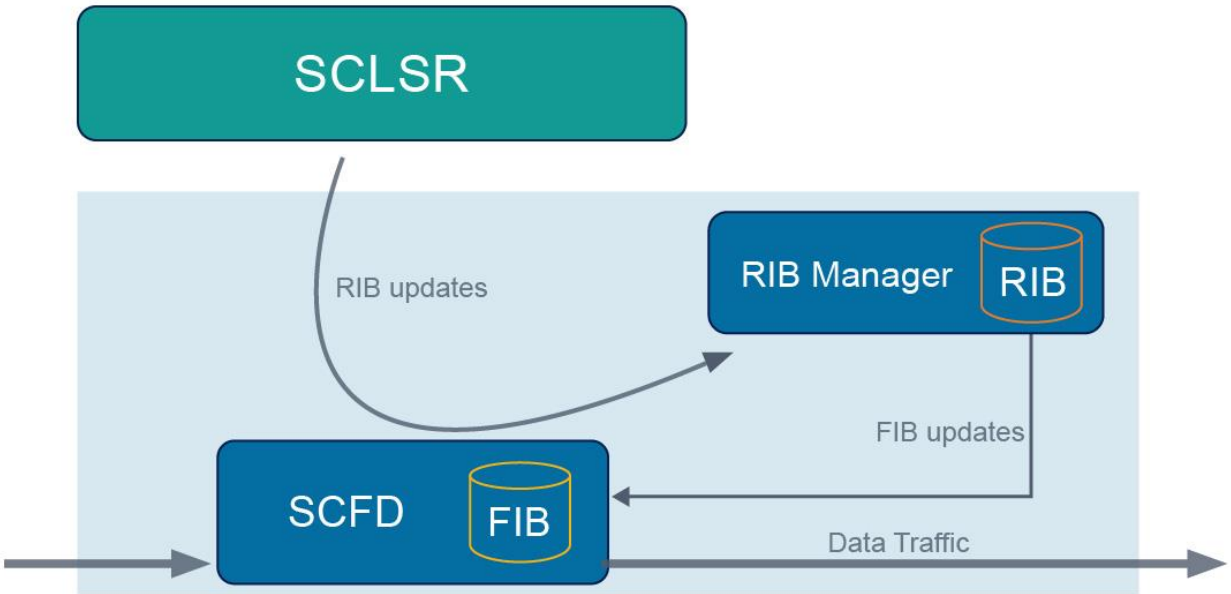
数据标识是基于区块网络命名规则层次化演进而来，不仅符合区块网络要求的命名规则，而且易于用户层的易读易理解。此外，信息的命名化还有利于应用层的领域编程。

3.5.1 分布式路由

根源链 BOS 的分布式路由是基于命名概念建立的。基于数据资产的安全性要求，通过部分继承命名链路状态路由协议，实现根源链 BOS 自主研发的分布式路由组件 SCLSR。此外分布式转发组件依赖于路由组件。

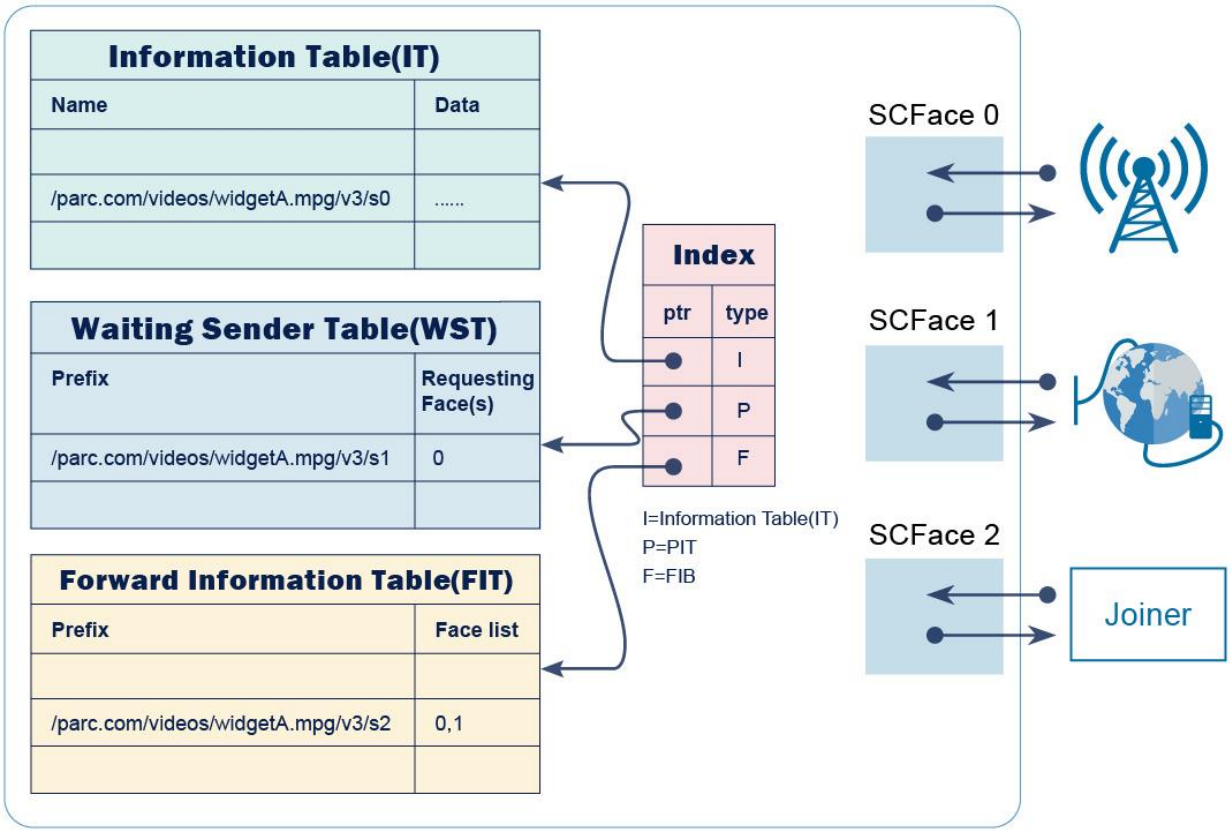
SCLSR 的主要设计目标是提供路由协议来填充区块网络的 FIT 表。SCLSR 使用链路状态或双曲线路由计算路由表，并为单个授权域中的每个可到达命名前缀生成多个 SCFace。

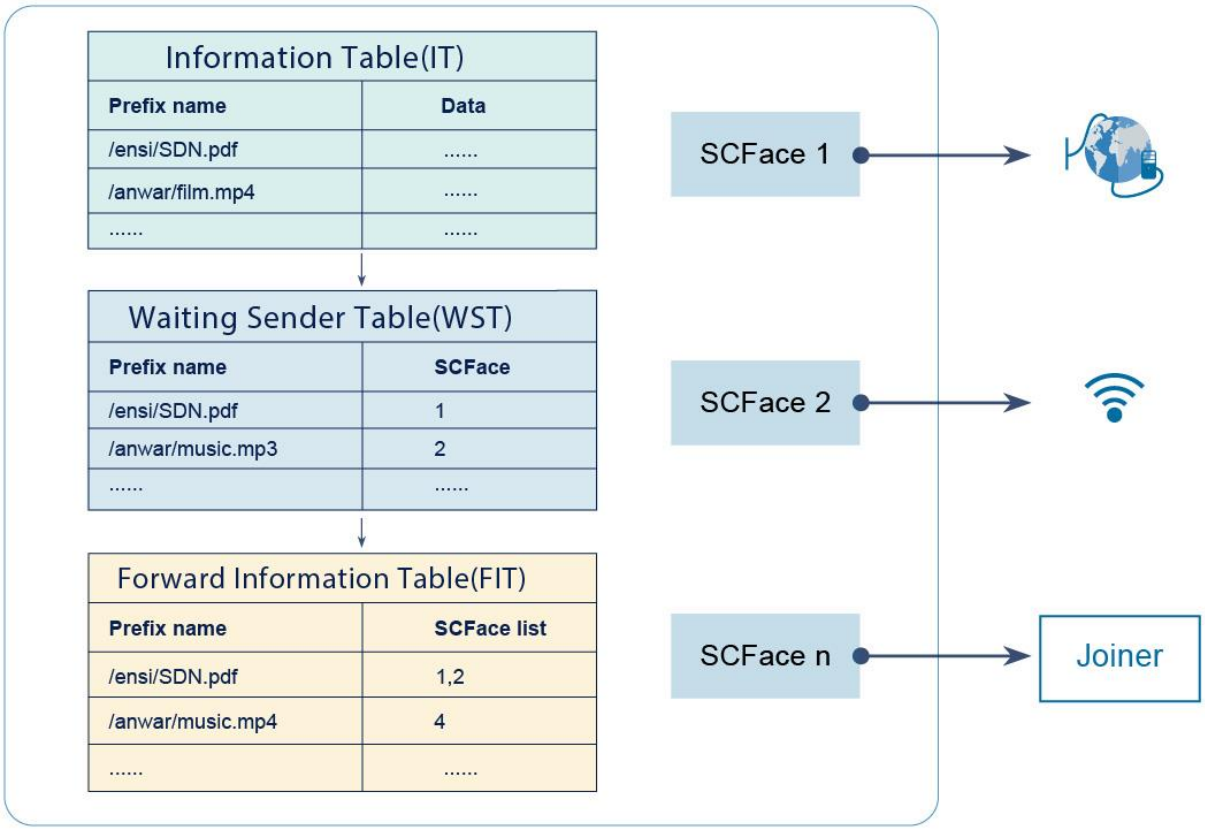
下图表示 SCLSR 构建 FIT 转发表的过程以及 SCLSR 与 SCFD 的关系：



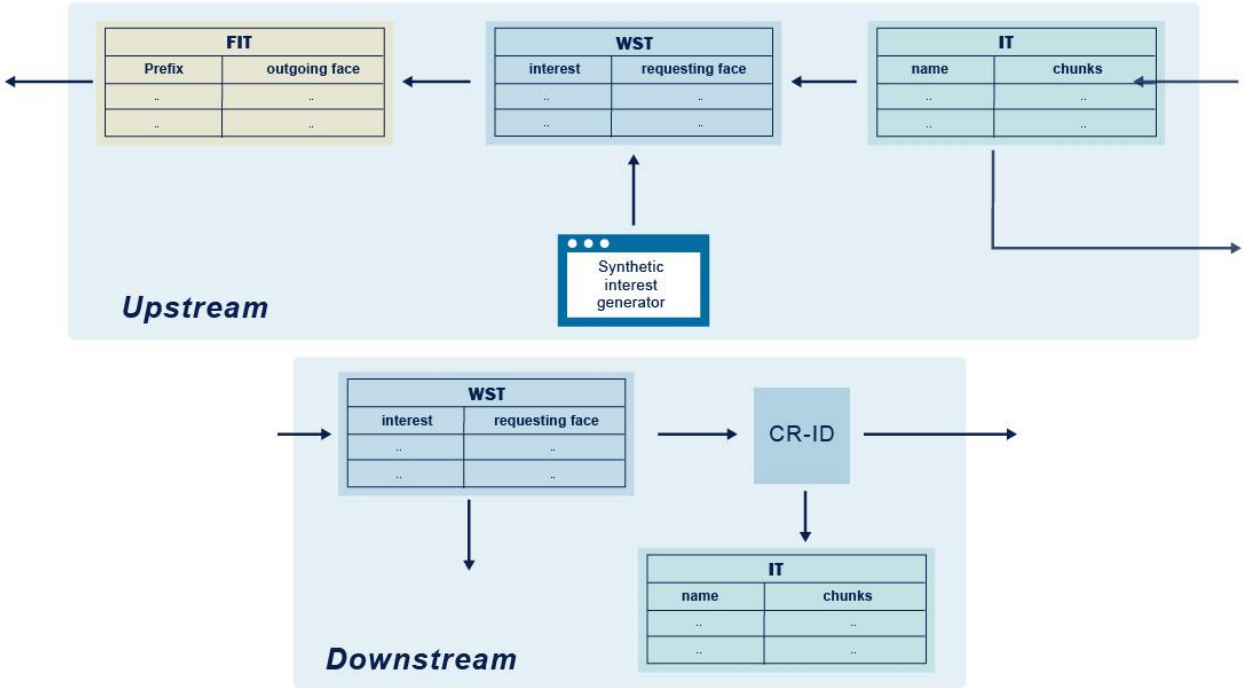
3.5.2 分布式转发

根源链 BOS 的分布式转发同样基于命名概念创建与工作。其中，FIT 表的构建依赖于 SCLSR，根源链 BOS 自主研发的分布式转发组件 SCFD 作为命名数据网络的核心实例，其内部构成如下图。





下图则说明了基于 SCFD 的上层数据发出(Upstream)，上层数据的接收(Downstream)逻辑。



3.5.3 分布式存储（Off-C)

根源链 BOS 分布式存储(Off-Chain)的底层存储基于 CEPH、BigChainDB，并引入了 BLFS（Block Label File System）技术机制。

在面向内容方面，CEPH 是用于存储内容可寻址文件的一种高可靠标准。内容可寻址存储是一种基于其内容而不是其位置来检索的信息存储机制。

根源链 BOS 重构了 CEPH 接入层的实现并结合分布式转发的命名规则，实现了基于内容的分布式存储(Off-Chain)服务实例。使用 CEPH 存储的所有文件名称都是从其内容的散列中生成的，并意味着同一个文件在每台计算机上都具有相同的名称，并且更改文件内容会导致文件名称的更改。当从服务器下载一个文件夹时,可以根据服务器提供的内容重新计算文件名称来验证文件是否为所请求的文件。CEPH 的 P2P 网络层,允许计算机根据其唯一的名称发现和共享文件。

面向检索，由于业务数据的规模庞大，数据存在查询检索的需要，根源链 BOS 链下存储服务实例又引入了 BigChainDB 作为业务数据的存储引擎。BigChainDB 具有去中心化控制、防篡改和创建传输数字资产等区块链技术的优点。防篡改通过几种机制实现:分片复制、不允许更新或修改、定期备份数据库、所有交易签名加密、区块和投票。任何有资产创建权的实体都可以创建一个资产，一个资产只有当新所有者满足加密条件时才可以被新所有者接收。这意味着黑客或者恶意管理员不能任意更改数据，而且没有单点错误风险。

另一方面，在面向内容时，根源链 BOS 基于区块链属性与特点引入了 BLFS 机制，当前公链基础设施在支持实际应用时，一大核心问题是数据存储如何解决。由于区块链是采用分布式存储方式，每个节点同步其他节点数据，即使经过优化，也面临着大量数据的存储问题。在结合了 BlockLabel 技术后，根源链引入了 BLFS 机制，形成了一种高效、高可靠的区块链形态分布式存储结构。BLFS 弥补了现有区块链系统在文件存储方面的短板，它为所有的区块链准备好了数据存储结构，可以链接到不同的区块链项目。用户可以使用 BLFS 来处理大量数据，然后把对应的加密哈希存储到区块链中并打上时间戳，这样就无需将数据本身放在链上，不但可以节省其他区块链的网络带宽，还可以对其数据进行有效保护，于是，BLFS 就成为了根源链这一公链基础设施中存储功能的实现方式。总之，链上(On-Chain)的 PoE 属性和链下(Off-Chain)的命名和内容规则共同组成了根源链 BOS 分布式存储。

3.6 加密层

根源链 BOS 的加密层主要包含三部分，第一是加密协议的实现，第二是硬件加密板卡的封装，第三是依赖加密协议实现和硬件加密之上的分布式加密服务组件。

3.6.1 分布式加密设计

基于分布式存储的需要，根源链 BOS 分布式加密组件引入了 KMS 密钥管理服务以及 Barbican。由于分布式存储是基于 CEPH 和 BigChainDB，所以必须依赖 KMS 和

Barbican 组件。数据资产的存储要求体现在安全方面在分布式机密组件中进行了重构和封装，并将这些接口暴露给 ASL 层。换言之，这些内容面向用户层面是透明的。

基于应用服务层 ASL 的需要，根源链 BOS 分布式加密组件提供了 SDK 与网关加密依赖。主要包括如下。为了保证数据资产的安全，P2P 节点之间的通信支持使用加密通道。现代信息安全技术所依赖的加密机制主要分为两种，一种是对称加密方式；一种非对称加密方式。两者在性能和应用场景上相互有区别。为了保证网络的安全性并且兼顾系统运行的性能，面向数据通讯采取对称加密机制；对称密钥的协商和传输采取非对称加密机制。

为了系统安全性及后续系统扩充方面考虑，根源链并未规定具体的加解密算法，而是参考 SSL 机制，实现了一套加密算法和加密密钥的 P2P 协商模块，用于 P2P 节点之间的通信协商。

3.6.2 对称加密机制

对称加密采用了对称密码编码技术，它的特点是文件加密和解密使用相同的密钥，即加密密钥也可以用作解密密钥，对称加密算法使用起来简单快捷，密钥较短，且破译困难，除了数据加密标准（DES），另一个对称密钥加密系统是国际数据加密算法（IDEA），它比 DES 的加密性好，而且对计算机功能要求也没有那么高。IDEA 加密标准由 PGP（Pretty Good Privacy）系统使用。为了更好的解释分布式加密的对称加密部分，如下几个问题值的注意。

- 要求提供一条安全的渠道使通讯双方在首次通讯时协商一个共同的密钥。直接的面对面协商可能是不现实而且难于实施的，所以双方可能需要借助于邮件和电话等其它相对不够安全的手段来进行协商；
- 密钥的数目难于管理。因为对于每一个合作者都需要使用不同的密钥，很难适应开放社会中大量的信息交流；
- 对称加密算法一般不能提供信息完整性的鉴别。它无法验证发送者和接受者的身份；

- 对称密钥的管理和分发工作是一件具有潜在危险的和烦琐的过程。对称加密是基于共同保守秘密来实现的，采用对称加密技术的贸易双方必须保证采用的是相同的密钥，保证彼此密钥的交换是安全可靠的，同时还要设定防止密钥泄密和更改密钥的程序。

常用的对称加密算法有 DES、3DES、Blowfish、IDEA、RC4、RC5、RC6 和 AES。

3.6.3 非对称加密机制

与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥（publickey）和私有密钥（privatekey）。公开密钥与私有密钥是一对，如果用公开密钥对数据进行加密，只有用对应的私有密钥才能解密；如果用私有密钥对数据进行加密，那么只有用对应的公开密钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加密算法。

非对称加密算法实现机密信息交换的基本过程是：甲方生成一对密钥并将其中的一把作为公用密钥向其它方公开；得到该公用密钥的乙方使用该密钥对机密信息进行加密后再发送给甲方；甲方再用自己保存的另一把专用密钥对加密后的信息进行解密。甲方只能用其专用密钥解密由其公用密钥加密后的任何信息。

非对称加密算法的保密性比较好，它消除了最终用户交换密钥的需要，但加密和解密花费时间长、速度慢，它不适合于对文件加密而只适用于对少量数据进行加密。

非对称加密的典型应用是数字签名。常见的非对称加密算法有：RSA、ECC、Diffie-Hellman、El Gamal、DSA（数字签名用）。根源链的区块链层使用的非对称加密实现是基于 DSA 的 ECDSA 算法。

3.6.4 加密协商模块

加密协商模块完成 P2P 节点两两之间的加密套件协商。需要协商的加密套件内容包含签名验证算法（RSA,ECDSA,SM2），数字信封算法（RSA,ECDH,SM2，NULL），哈希、MAC 算法（SHA256，HMAC,SM3），对称加密算法（AES-CBC,AES-ECB,SM4，NULL）

其中括号内为目前支持的算法，包括美标和国密两种标准，更多的算法还在继续扩展中。其中 NULL 代表不加密或者不用数字信封。

这部分是协议层加密协议的实现，而且也是应用层 SDK 和网关 API 加密部分的基础服务组件。

3.6.5 密钥及证书管理

证书中存储着用户的公钥。公钥用来对签名的验证，也可以用来做数字信封。公钥证书管理证书的验证，证书中公钥的管理和使用。

- 私钥管理，非对称加密中私钥的管理包括私钥的生成、使用、删除以及导入、导出。
- 数字信封，对称密钥从发送方给接受方需要用到数字信封。数字信封确保加密密钥不被第三方窃取。
- 对称加密密钥管理，加密的双方密钥的生成、使用、删除。可以根据加密协商模块协商出来的对称加密算法，生成不同算法的密钥。

3.6.6 压缩传输支持

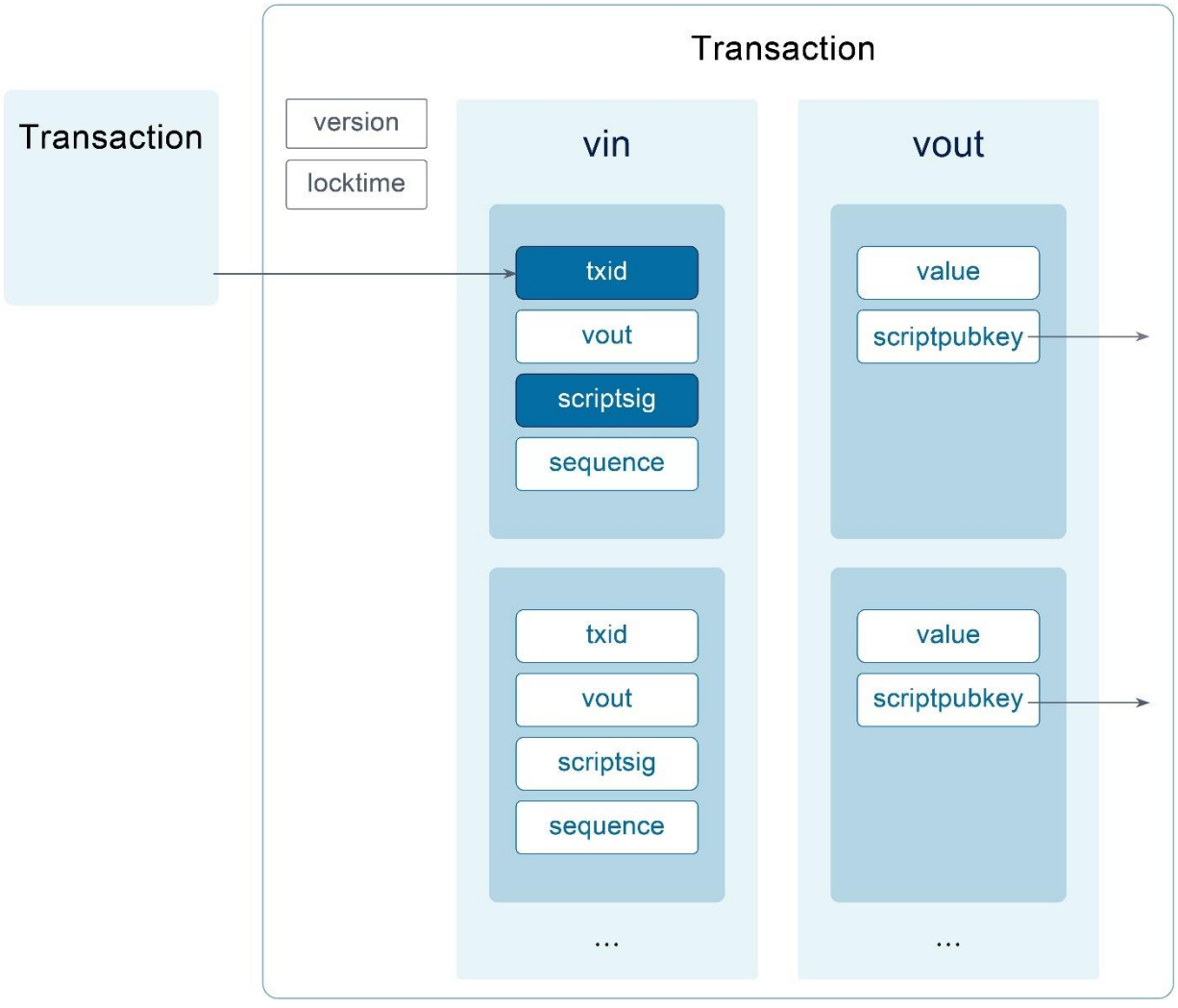
为了提高整个系统的吞吐量，传输的时延以及网络可靠性将很大程度上影响整个系统的稳定性。因此我们引入了压缩机制，采取压缩算法能大幅度降低节点之间数据资产和区块传输的数据量大小。

节点之间的数据压缩算法和加密协商一致，在加密协商时节点协商自身的压缩能力（7z，zip，NULL）。

3.7 隔离见证

隔离见证（segregated witness，简称 segwit，缩写 SW）如果摆脱根源链来讲就显得空洞，隔离即分离，将 A 分离成 B 和 C；而见证则可以理解为一个加密难题的解决方案。这是加密学角度的阐释。

在根源链上下文中，一个数字签名就是一种类型的见证。更确切地说见证是一个解锁脚本。



scriptsig 就是解锁脚本，作用是解除 txid 的锁定。其中 txid 的锁定体现在 scriptpubkey，锁定的对象是 value，也就是资产。

3.7.1 使用 SW 的原因

如果不限制区块的大小，就会受到 DDOS 攻击。DDOS 攻击是通过伪造区块阻塞网络来达到攻击的目的。试想每个用户的带宽是有限的，如果区块大小不受限制，那么正常的、异常的区块通过网络来往于每个用户，不仅用户的客户端需要处理大规模的数据，而且网络负载也相当巨大，最终导致客户端消耗大量的时间处理无用的伪造区块，而无法处理正常区块的业务；

方案：限制区块大小到 1MB。通过限制区块大小，那些伪造的大区块很快被检测并被丢掉。

而虽然上述方案解决了 DDOS 问题，但是却引入了新的问题：容量不足。

随着根源链的关注和使用的提高，1MB 的区块将可能出现瓶颈，这主要表现在：

- 单个交易的增大，而且主要是签名部分数据量大(问题 1)

- 单个区块打包交易数量有限，导致交易效率极低(问题 2)

- 由于签名算法强壮度问题，从而通过签名伪造导致 TXID 被伪造。虽然面向区块链网络没有多少影响，但是这种攻击会导致第三方交易平台资产的损失。这种攻击也被称之为延展性攻击(Malleability Attack)(问题 3)

目前主流的处理方法主要是 3 个方面：

[1] 提高区块容量

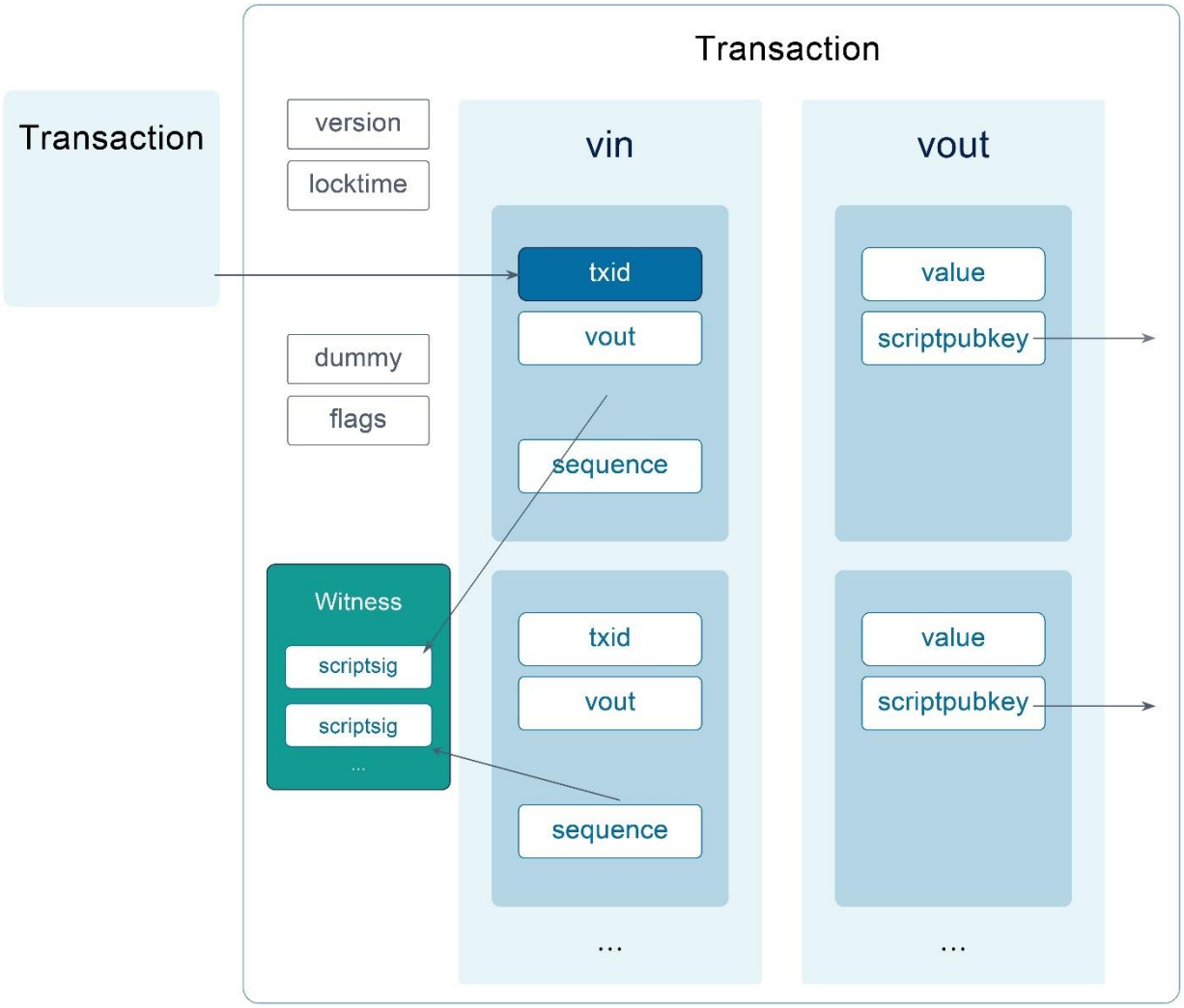
[2] 使用附加结构，例如闪电网络(Lightning Network，LN)

[3] 在现有的结构中优化，例如隔离见证(SW)

方案：通过问题 123 的反映，分离签名的方式更为托贴，也就是隔离见证技术。此外，闪电网络(LN)也是基于隔离见证(SW)之上构建的。

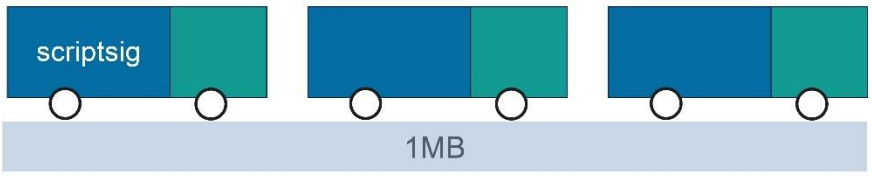
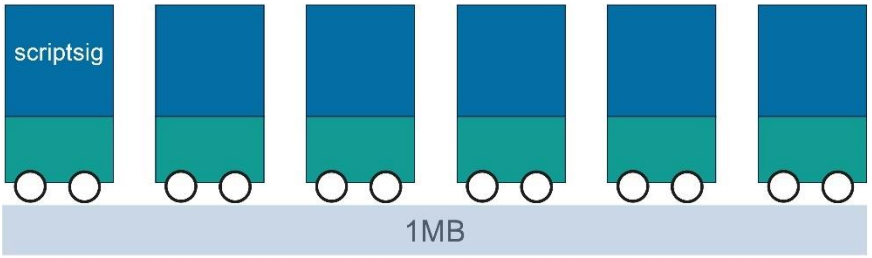
3.7.2 隔离见证 SW 优点

隔离见证后的交易结构如下，注意：scriptsig 被剥离出来，通过新增 flags 控制 Witness 部分的去留。



3.7.3 结构与算法变化

隔离见证后的区块结构也发生了变化。

没有激活隔离见证的交易与区块之间的组织关系	
激活隔离见证的交易与区块之间的组织关系	

隔离见证后的区块组织方式导致了新的计算规则，最直接的变化就是区块原本的上限使用 1MB(M Bytes)进行度量，现在使用 4M WU(Weight Unit)进行度量。

因此，基于隔离见证技术的区块扩容方式，属于技术扩容，而不是物理扩容。由于涉及内容较多，细节不赘述。

3.7.4 优势

- 彻底解决了延展性攻击(Malleability Attack)问题，因为隔离(剥离)了见证数据之后的交易被创建是无法变更的。

- 手续费(Fee)的减少，因为手续费取决于两个要素：[1] 单笔交易的容量 [2] 单位容量的手续费定价。公式：Fee = 单笔交易容量 x 单位容量手续费定

价。例如，Fee = 2KB x 0.003 BSTK/KB。由于隔离见证后，每笔交易剥离了见证数据，所以单笔交易的容量减少。

- 区块打包交易增多。
- 交易确认性能提高，因为区块打包交易增多。隔离见证前，某交易可能需要等待若干确认期之后，才能从内存池打包到区块中；隔离见证后，交易被打包的概率明显升高。
- 众多技术的基础，例如闪电网络、侧链、Omni 的实现均基于 SW 软分叉，即向后兼容，用户使用的软件客户端允许不升级到具有隔离见证能力的新版本；但我们建议每个用户都升级，因为能够支撑闪电网络等新技术，带来更多用户体验。

3.8 侧链

侧链的出现是为了解决不同类型区块链实例和不同类型数字货币实例的对接问题。根源链 BOS 将侧链技术规划为协议部分和实现部分，其中协议部分主要是指 2WPP 双向锚定协议的自主化定义，不仅包含了业界侧链的规范约定，而且包含了根源链 BOS 的内部多链侧链的协议规范内容。

3.8.1 侧链协议层与 2WPP 双向锚定协议

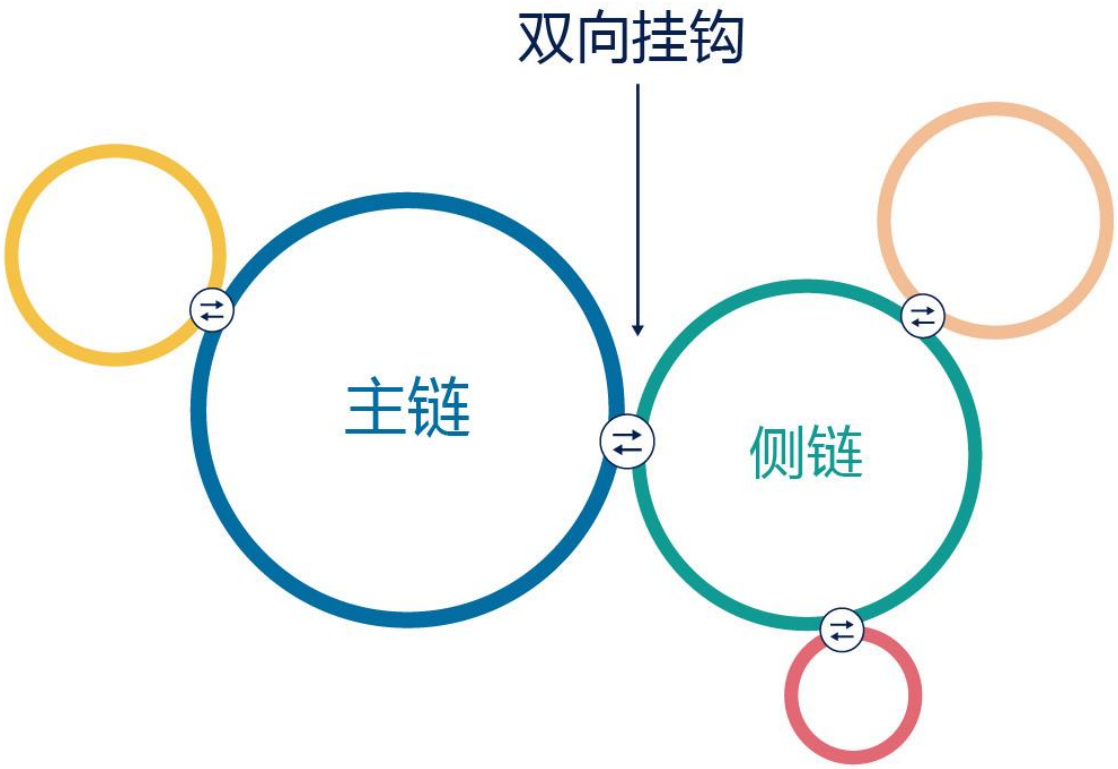
在协议层，首先约定主链实例（Parent chain instance），然后约定其他类型区块链实例作为侧链，二者通过双向锚定（2WPP），可实现主链到侧链，侧链到主链进行流通的需要。

侧链可以是一个独立的区块链，有自己按需定制的账本、共识机制、交易类型、脚本和合约的支持等。侧链不能发行 Token，但可以通过支持与主链 Token 挂钩来引入和流通一定数量的 Token。当主链 Token 在侧链流通时，主链上对应的 Token 会被锁定，

直到这些 Token 从侧链回到主链。可以看到，侧链机制可将一些定制化或高频的交易放到主链之外进行，实现了区块链的扩展。

侧链的核心原理在于能够冻结一条链上的资产，然后在另一条链上产生，可以通过多种方式来实现。

根源链 BOS 的 2WPP 协议的设计难点在于如何让资产在主链和侧链之间安全流转。简言之，接受资产的链必须确保发送资产的链上的 Token 被可靠锁定。



3.8.2 侧链实现层

2WPP 协议要求采用双向锚定机制实现主链向侧链转移和返回。主链和侧链需要对对方的特定交易做 SPV 验证，因此实现 SPV 验证是侧链实现层的主要内容。

简单支付验证（Simplified Payment Verification，SPV）能够以较小的代价判断某个交易是否已经被验证过（存在于区块链中），以及得到了多少算力保护（定位包含该交

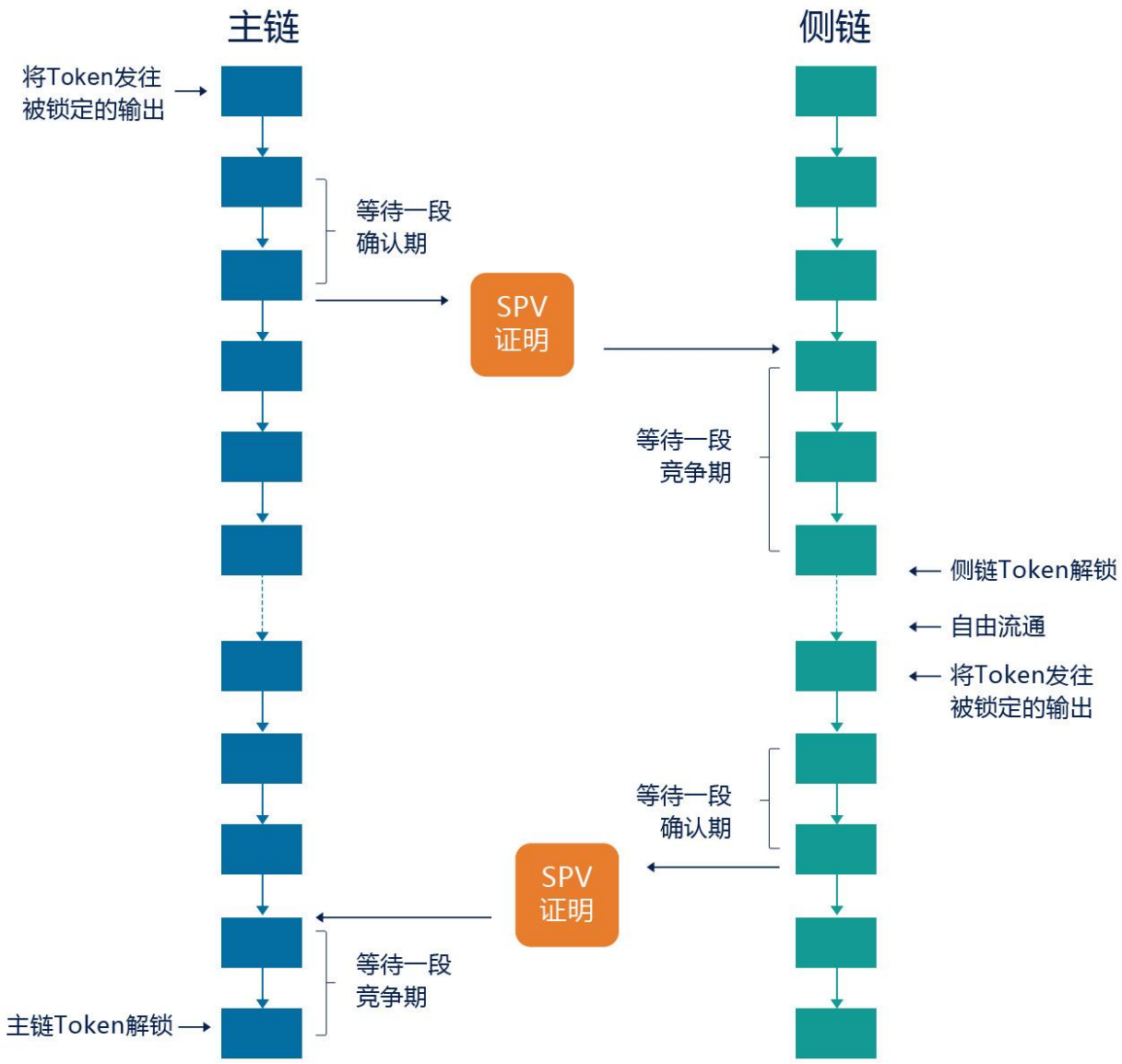
易的区块在区块链中的位置）。对方链(主链或者侧链)只需要下载所有区块的区块头（Block Header），并进行简单的定位和计算工作就可以给出验证结论。

2WPP 协议中，用 SPV 来证明一个交易确实已经在区块链中发生过，称为 SPV 证明（SPV Proof）。一份 SPV 证明的文本结构包括两部分内容：一组区块头的列表，表示工作量证明；一个特定输出（output）确实存在于某个区块中的密码学证明。

3.8.3 SCA 侧链适配器

根源链 BOS 的 SCA 侧链适配器的完整实现过程如下

- 事件发生端要向侧链转移 Token 时，首先在主链创建交易，待转移的 Token 被发往一个特殊的输出。这些 Token 在主链上被锁定。
- 等待一段确认期，使得上述交易获得足够的工作量确认。
- 事件发生端在侧链创建交易提取 Token，需要在这笔交易的输入指明上述主链被锁定的输出，并提供足够的 SPV 证明。
- 等待一段竞争期，防止双重花费攻击。
- Token 在侧链上自由流通。
- 当事件发生端想让 Token 返回主链时，采取类似的反向操作。
- 首先在侧链创建交易，待返回的 Token 被发往一个特殊的输出。
- 先等待一段确认期后，在主链用足够的对侧链输出的 SPV 证明来解锁最早被锁定的输出。
- 竞争期过后，主链 Token 恢复流通。



3.8.4 根源链侧链技术栈

-侧链（SideChain）与 2WP（2 Way Peg）

-联合楔入（Federated Peg）

-单一托管（Single Custodial）

-简单支付验证证明（Simple Payment Verification Proof，SPV Proof）

-联合挖矿（Federated Mining）

-驱动链（DriveChain）

-交易压缩协议网络（Lumino Transaction Compress Protocol Network，LTCP Network）

3.8.5 侧链实现细节

侧链是附生在主链之上的区块链网络。侧链与主链的角色能够互换，属于相对概念。例如，目前典型的主链是底层基于比特币系统的网络。

面向主链层面：

-主链轻易不能变更，太多尝试和创新需求就需要放在侧链实现

-主链软硬分叉的代价过高，分布式系统上线之后往往维护难度远远高于中心化系统

-主链现行规则和协议不能打破

面向市面上其他 Token 层面：

-创新少，投机多

-链生态碎片化，难以统一

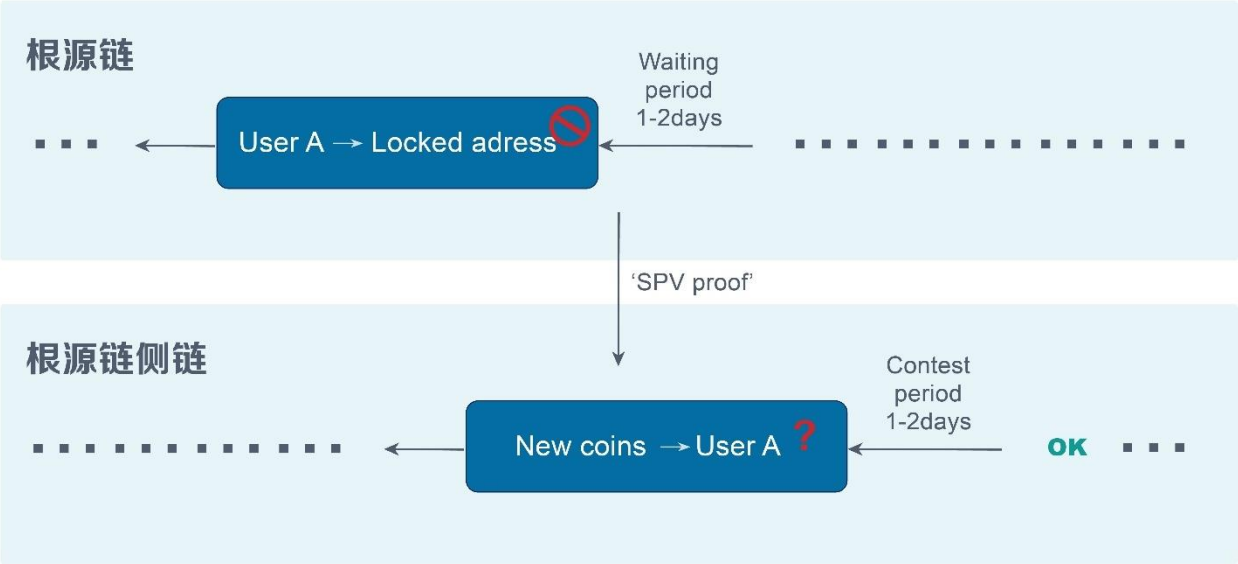
-缺乏验证机制、缺乏安全、缺乏数量和质量上的交易所

面向其他技术：

-基于比特币区块链技术的类侧链，例如万事达币 Omni/MasterCoin，合约币 Counterparty（XCP），基于 Omni 实现的 TetherCoin 等，但都不是侧链

-非比特币区块链技术的类侧链，例如染色币 ColoredCoin 等，纯粹只是应用层实现

如下图，Parent Chain 代表主链，SideChain 代表侧链，主链上 A 把币锁到特殊地址，然后确认若干区块，发送 SPV proof 到侧链，侧链收到 SPV proof 之后，创建侧链币指向 A 在侧链的地址，然后在侧链确认若干个区块。



-单向楔入。单向楔入是一个过程，是指从链的一侧锁定资产，经过“等待期”之后，向另一链发送 SPV Proof，然后创建该侧链的等价资产，经过“竞争期”之后，该用户该侧资产就可以被激活可用。

-SPV Proof。简单支付验证证明基于区块链的区块头和 PoW Proof 获得。包括 PoW Proof 的确认高度，锁定资产的交易对应的区块的块头信息（Merkle root，锁定交易到根的构造路径 Hash 值，Nonce 值，上个区块 PreHash 值）

单向楔入问题：

-为了支持单向楔入侧链实现，网络设计变的复杂

-钱包变的复杂，要求支持多币种

-趋于网络中心化

-无法避免主链软分叉风险

-侧链被 51%攻击很容易 例如偷币，银行挤兑，使用者不愿承担侧链损失难以推广等问题

方案：

-针对攻击问题，侧链可采用联合挖矿（Federated Mining），篇幅有限，不做赘述

-针对安全问题，侧链可以采用 zk-SNACK，也就是 ZCOIN 引入并实现的加密学技术

3.8.6 双向楔入实现

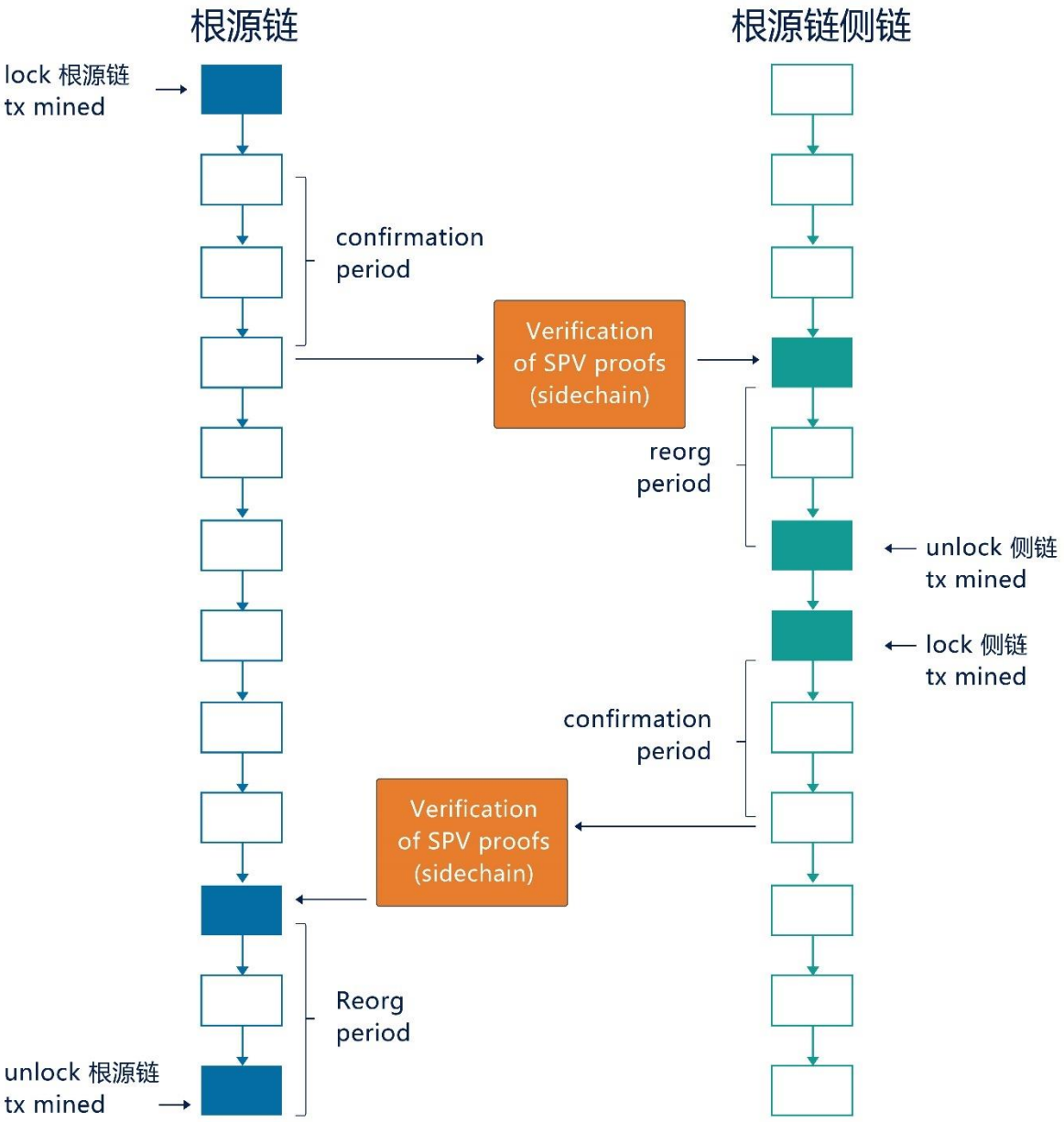
双向楔入分为四个阶段，具体如下：

-发送锁定交易，把根源链锁定在根源链主链上。

-等待确认期。确认期的作用是等待更多区块确认锁定交易，可防止假冒锁定和拒绝服务攻击，等待时间是 1～2 天，或根据程序固定的时间。

-在侧链上赎回根源链。上述确认期结束后，用户在侧链上创建一个交易花掉锁定交易的输出，并且提供一个 SPV 工作量证明，输出到自己侧链上的地址。这个交易也称为赎回交易，SPV 工作量证明是指赎回交易所在区块的工作量证明。

-等待一个竞争期。竞争期的作用是防止双重支付，在此期间，新转移过来的币不能在侧链上花费。竞争期的目的是防止重组时出现双花，在重组期间会转走先前锁定的币。在这个延迟期内的任何时刻，如果有一个新的工作证明发布出来，且对应的有着更多累计工作量的链中没有包含那个生成锁定输出的区块，那么该转换将被追溯为失效。我们称此为重组证明。



竞争期结束后，这个赎回交易将被打包到区块中，用户就可以使用自己的根源链了，从侧链转移根源链的过程也是如此，当用户想把币从侧链上转回父链时，与原先转移所用的方法相同：在侧链上将币发送至一个 SPV 锁定的输出，并产生一个充分的 SPV 证明来表明该输出已经完成，然后使用这个证明来解锁父链上先前被锁定的那个等面值的输出。

由于楔入式侧链可能会从很多链中搬运资产，且无法对这些链的安全性做出假定，因此，不同资产不可相互兑换是非常重要的（除非是一个显式声明的交易）。否则，恶意用

户可以通过创建一条资产毫无价值的无价值链进行偷盗，即将这样一种资产移到一个侧链，再用它去兑换别的东西。为了应对这种情况，侧链必须有效地将不同父链中的资产处置为不同的资产类型。

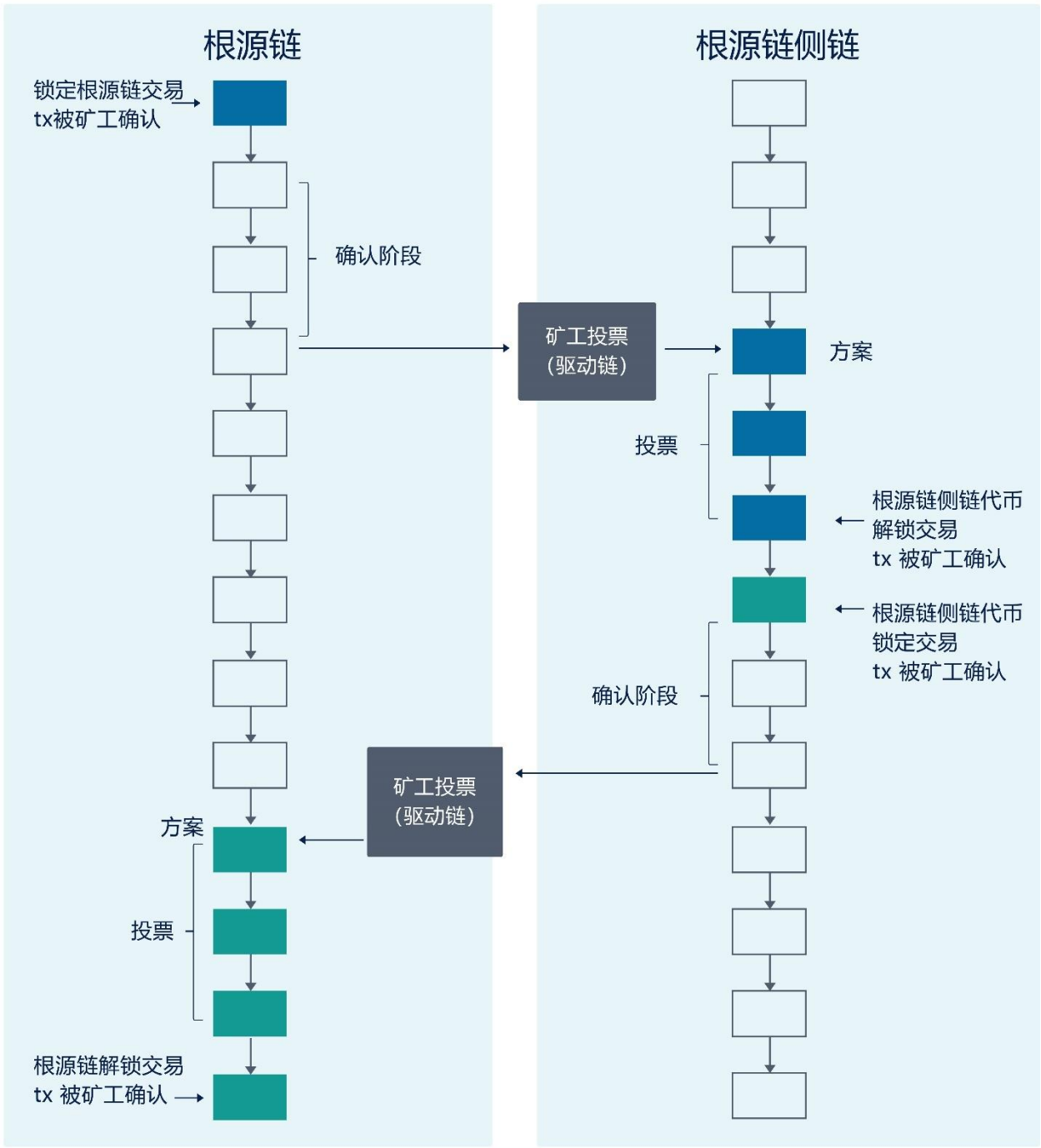
总之，这一设计的意图是让父链和侧链相互做数据的 SPV 验证。由于不能期望父链客户端能看到每条侧链，因此为了证明所有权，用户必须从侧链导入工作量的证明到父链。在对称式双向楔入中，反向的操作也是如此。

为了让根源链系统成为相对应的父链，需要有一个能识别和验证 SPV 证明的脚本扩展。最起码的要求是，这种证明需要做得足够小，以便能放进根源链系统一个交易之中。不过，这只是一个软分叉，对于不使用新功能的交易不会产生影响。

3.8.7 驱动链实现

使用驱动链是因为主链如果是根源链区块链系统，或者是根源链区块链叉出的新型公链系统，此时实现双向锚定会复杂化。因此主链的变动最好没有或很小；主链的变更会带来很大的风险与代价。

驱动链将被锁定根源链的监管权交给根源链矿工，并且允许根源链矿工们投票决定何时解锁根源链和将解锁的根源链发送到哪里。矿工使用根源链区块链投票，使用区块里的某些字段来实现投票(例如 coinbase 字段)。越多的诚实矿工参与进来投票,则安全性就越高。下图是对驱动链的描述。


根源链
锁定根源链交易
tx被矿工确认
确认阶段
矿工投票
(驱动链)
方案
投票
根源链侧链代币
解锁交易
tx 被矿工确认
根源链侧链代币
锁定交易
tx 被矿工确认
确认阶段
矿工投票
(驱动链)
方案
投票
根源链解锁交易
tx 被矿工确认
根源链侧链
受影响的区块中的根源链侧链 ->根源链转换
受影响的区块中的根源链 ->根源链侧链转换

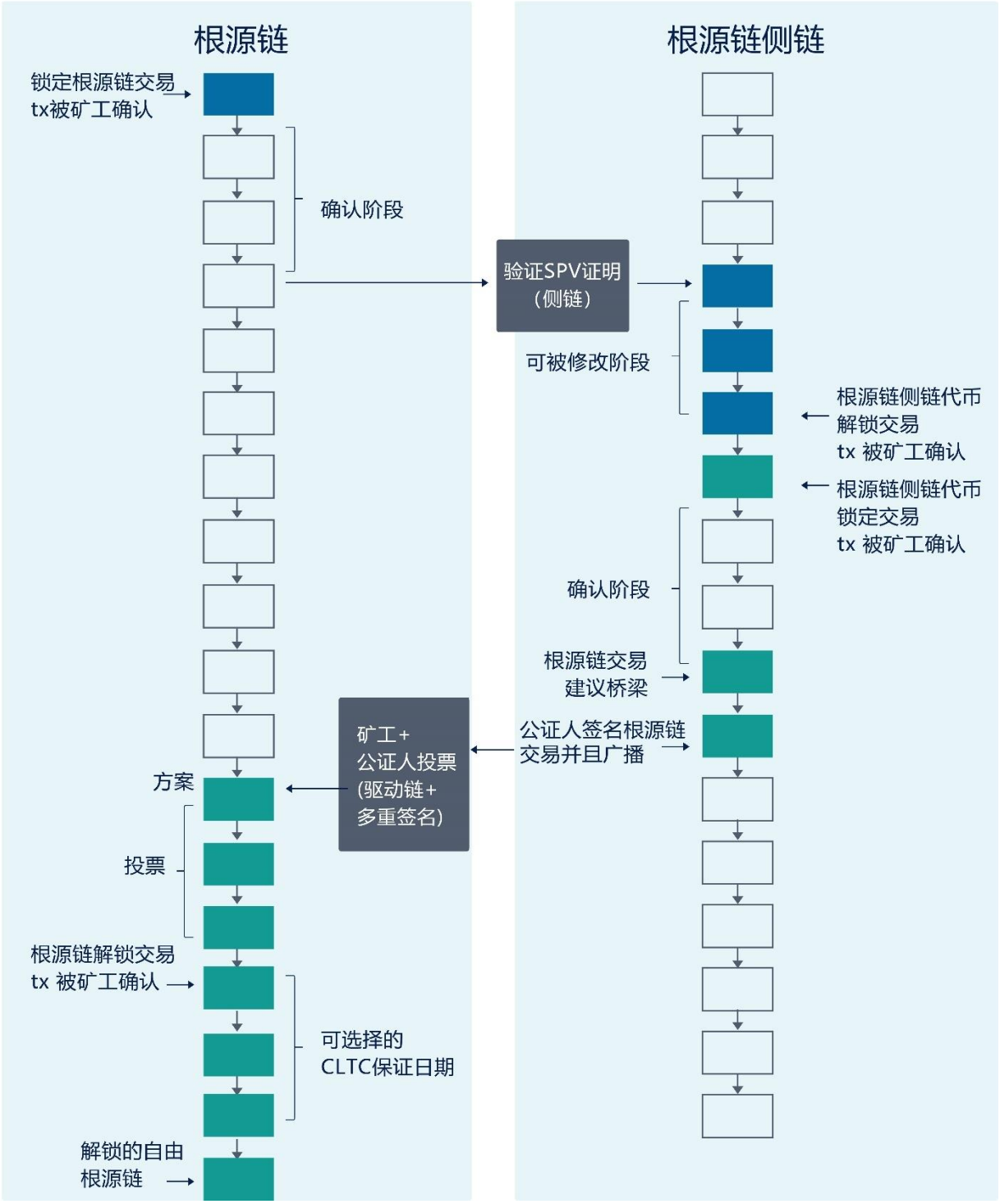
易压缩协议网络

交易压缩协议网络（Lumino Transaction Compress Protocol Network，LTCP Network）是类似于 Lightning Network 的技术，Lumino 是一种新型的链下支付渠道网络。

使用 LTCPN 的原因是，如果闪电网络覆盖数十亿用户，仍旧产生大量的链上交易。每当一个中心出现问题或者当一方消失的时候，这样的情况就会发生。此外，用户需要经常给他们的支付通道补充余额，这就需要更多的链上交易。假设每个用户每个月都使用一个主要的支付通道，那么每个通道每个月也需要结算，并且要求不会出现重大问题的节点，这样吞吐量才能够容纳数百万用户使用支付通道。总之，当基数飙升后，闪电网络的压力依旧很大，依旧不能根本解决性能问题。

LTCP 方案只占用区块 5 字节的空间来完成余额更新，将能够容纳数十亿级用户使用 BSTK 侧链系统。技术细节不赘述。

3.8.8 根源链侧链实现汇总



当矿工参与联合挖矿的程度比较低时，“驱动链/侧链”的安全性是相对低的。为了保证整个网络安全运行，采用双向锚定在侧链，驱动链在主链的混合设计方式；其设计要点分为三点：

-主链的安全性是基于驱动链外加一组公证人。

-矿工和公证人(拥有不同的权重)共同投票决定解锁哪些主链资产。

-公证人使用数字签名进行投票，而矿工则在他们的 coinbase 交易中写入一个特殊的标记进行投票。

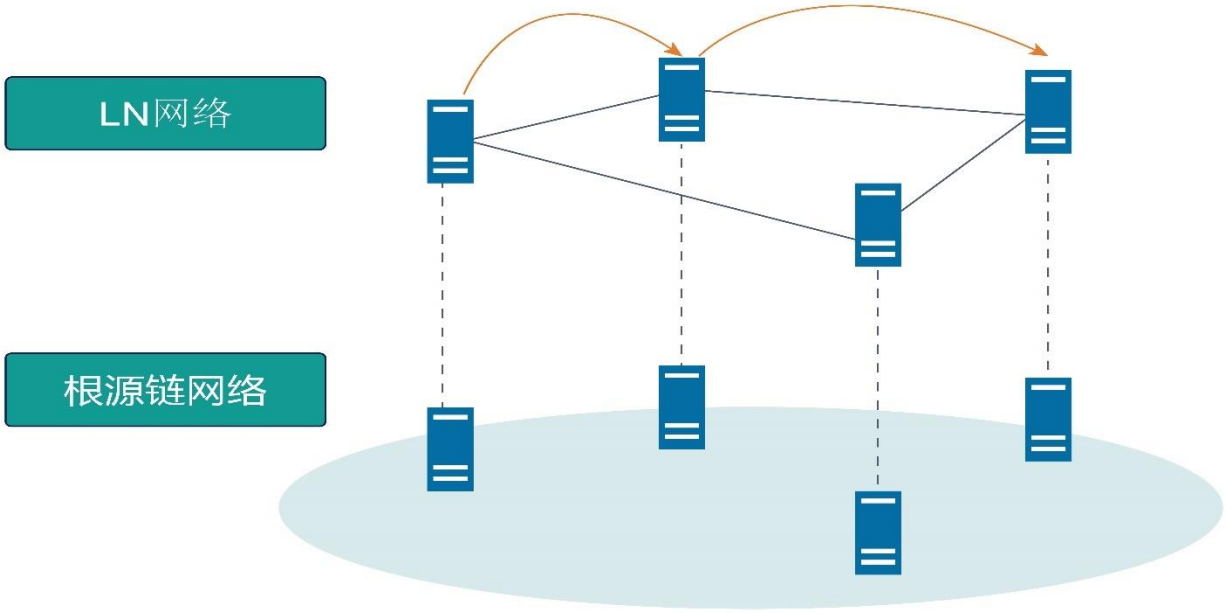
因此，这是中心化和安全性之间的一种权衡博弈。最终的“类双向锚定”设计可以被称为“驱动链+公证人+2WP”的侧链设计。

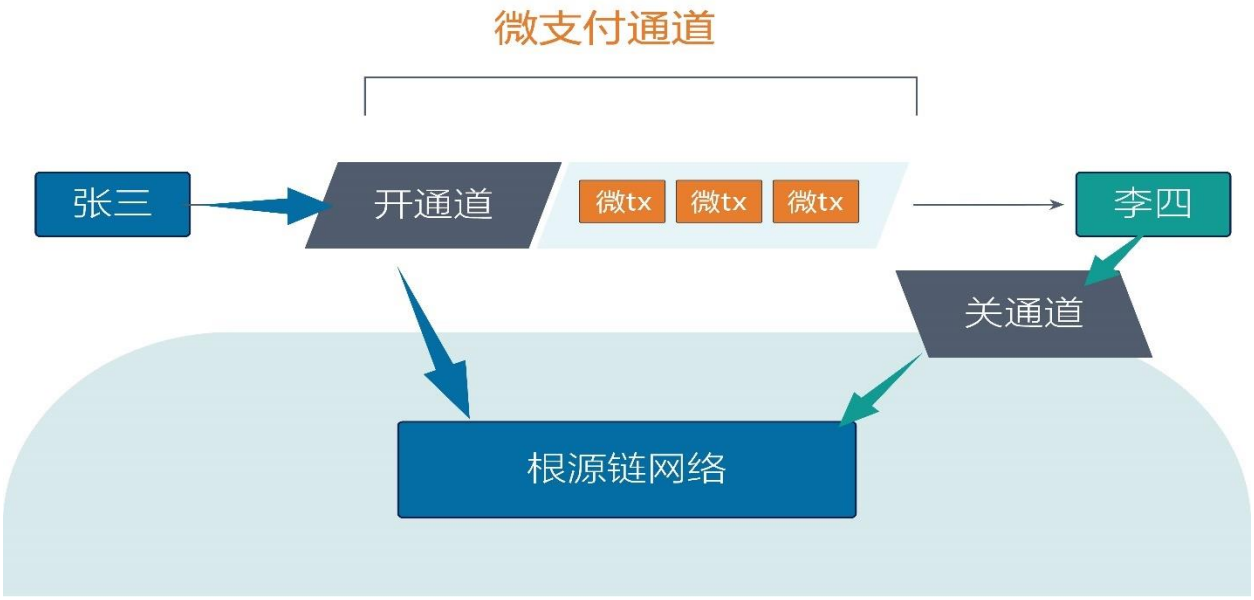
3.9 闪电网络

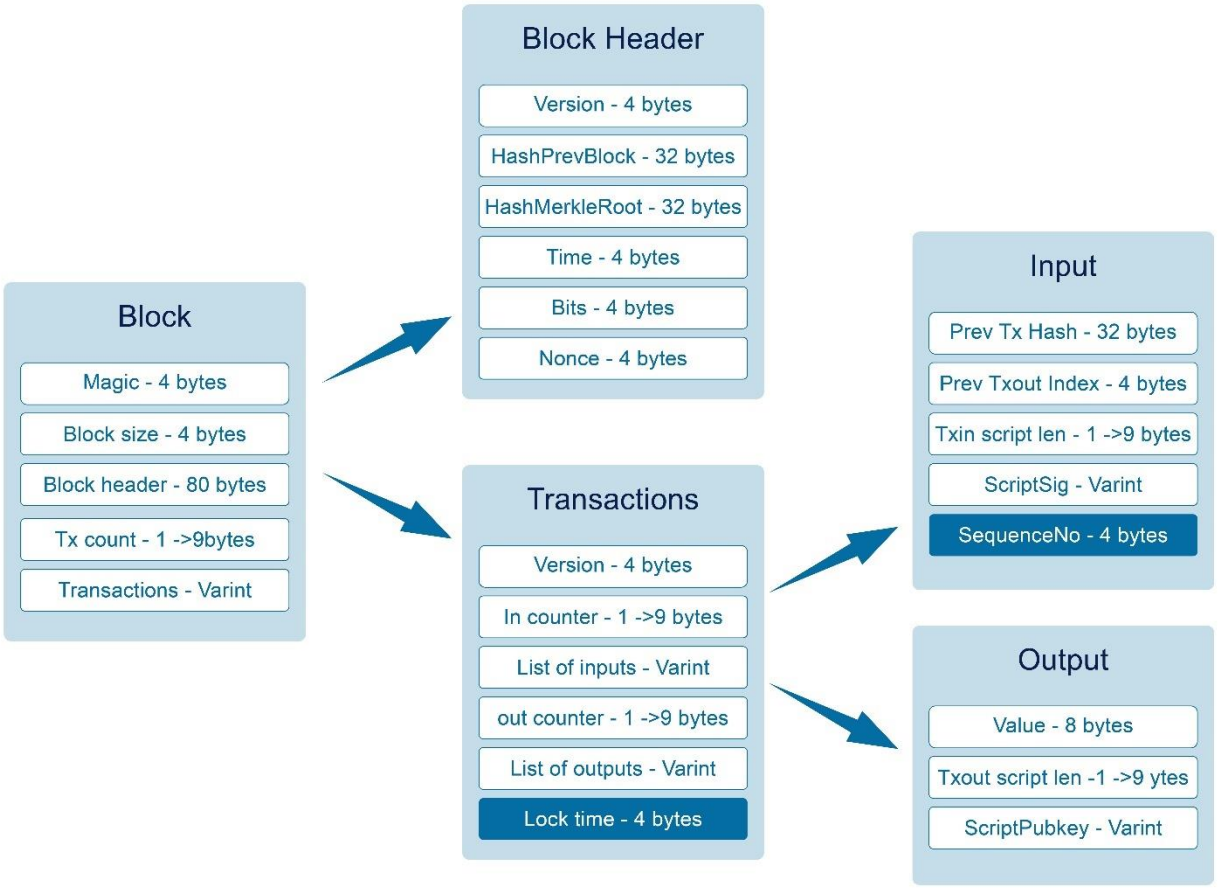
闪电网络（Lightning Network，LN）从软件层面来看是一个去中心化的系统。而应用层面，LN 则可视为一个建立在根源链之上的系统，它可以让人们立即发送/接收付款，并通过让他们远离主网络(根源链)来降低交易费。在技术层面上，LN 是基于微支付通道，设计出两种类型的交易合约：序列到期可撤销合约(Revocable Sequence Maturity Contract，RSMC)，哈希时间锁定合约(Hashed TimeLock Contract，HTLC）。

RSMC 解决了微支付通道单向的缺陷，以及通道两端任何一方恶意对待交易的问题。HTLC 解决了 RSMC 无法跨节点微支付的问题。其中 RSMC 与 Transaction 的每个 Sequence 字段有关。HTLC 与 Transaction 的 nLockTime 字段有关。通过利用 Sequence 与 nLockTime 构建出来的交易则具备了合约能力。其中 nLockTime 就是锁定时间。

因此从技术层面上，两个类型的交易组合构成了闪电网络。


LN网络
根源链网络


微支付通道
张三
开通道
微tx
微tx
微tx
李四
关通道
根源链网络



3.9.1 为何引入闪电网络

潜在问题 1：如果频繁的使用根源链网络主网，不论大小交易，均需要根据数据量大小收取交易费。

方案：通过构建微支付通道，只有开启通道和关闭通道两次需要与根源链网络接触，其余在通道开启过程中的若干笔交易的增删改与根源链网络无关。

潜在问题 2：根源链的交易确认需要一定时间。

方案：LN 的通道两端结算是瞬间的，而且 LN 设计的交易类型具备抵赖行为，因而通道双方去信任(即交易双方不需要信任)。

潜在问题 3：隐私问题如何改善。

方案：因为开启通道和关闭通道两次需要与根源链网络接触，所以原本每笔交易都需要上链，现在只有两笔上链，主网追溯概率降低。

潜在问题 4：不开启隔离见证(SW)区块 1MB 上限，开启隔离见证区块 4M WU 上限，在业务频繁作用于主网的场景下，区块容量不足的压力仍旧巨大。

方案：因为开启通道和关闭通道两次需要与根源链网络接触，所以海量的微小交易不会对现有区块容量构成压力威胁。

3.9.2 根源链时间锁定技术

时间锁是只允许在一段时间后才允许支出的交易。根源链从一开始就有一个交易级的时间锁定功能。它由交易中的 nLocktime 字段实现。根源链推出了两个新的时间锁定功能，提供 UTXO 级别的时间锁定功能：CHECKLOCKTIMEVERIFY 和 CHECKSEQUENCEVERIFY。

时间锁对于后期交易和将资金锁定到将来的日期很有用。更重要的是，时间锁将根源链脚本扩展到时间的维度，从而能够支持复杂的多级智能合约。

3.9.2.1 交易锁定时间（nLocktime）

根源链从一开始就有一个交易级的时间锁功能。交易锁定时间是交易级设置（交易数据结构中的一个字段），它定义交易有效的最早时间，并且可以在网络上中继或添加到区块链中。

锁定时间也称为 nLocktime，是来自于根源链 Core 代码库中使用的变量名称。在大多数交易中将其设置为零，以指示即时传播和执行。如果 nLocktime 不为零，低于 5 亿，则将其解释为块高度，这意味着交易无效，并且在指定的块高度之前未被中继或包含在块链中。

如果超过 5 亿，它被解释为 Unix 纪元时间戳（自 Jan-1-1970 之后的秒数），并且交易在指定时间之前无效。指定未来块或时间的 nLocktime 的交易必须由始发系统持有，并且只有在有效后才被发送到根源链网络。如果交易在指定的 nLocktime 之前传输

到网络，那么第一个节点就会拒绝该交易，并且不会被中继到其他节点。使用 nLocktime 等同于一张延期支票。

nLocktime 就是一个限制，虽然它可以在将来花费，但是到现在为止，它并不能使用它们。我们来解释一下，下面的例子。

User A 签署了一笔交易，支付给 User B 的地址，并将交易 nLocktime 设定为 3 个月。User A 把这笔交易发送给 User B。有了这个交易，User A 和 User B 知道：

- 在 3 个月过去之前，User B 不能完成交易进行变现。
- User B 可以在 3 个月后接受交易。

然而：

- User A 可以创建另一个交易，双重花费相同的输入，而不需要锁定时间。 因此，User A 可以在 3 个月过去之前花费相同的 UTXO。
- User B 不能保证 User A 不会这样做。

了解交易 nLocktime 的限制很重要。 唯一的保证是 User B 在 3 个月过去之前无法兑换它。 不能保证 User B 得到资金。 为了实现这样的保证，时间限制必须放在 UTXO 本身上，并成为锁定脚本的一部分，而不是交易。

这是通过以下形式的时间锁定来实现的，称为检查锁定时间验证(CLTV)。

3.9.2.2 检查锁定时间验证 Check Lock Time Verify (CLTV)

根源链引入了一种新形式的时间锁进行根源链软分叉升级。根据 GYLIP-65 中的规范，脚本语言添加了一个名为 CHECKLOCKTIMEVERIFY（CLTV）的新脚本操作符。CLTV 是每个输出的时间锁定，而不是每个交易的时间锁定，与 nLocktime 的情况一样。这允许在应用时间锁的方式上具有更大的灵活性。 简单来说，通过在输出的赎回脚本中添加 CLTV 操作码来限制输出，从而只能在指定的时间过后使用。

当 nLocktime 是交易级时间锁定时，CLTV 是基于输出的时间锁。

CLTV 不会取代 nLocktime，而是限制特定的 UTXO，并通过将 nLocktim 设置为更大或相等的值，从而达到在未来才能花费这笔钱的目的。

CLTV 操作码采用一个参数作为输入，表示为与 nLocktime（块高度或 Unix 纪元时间）相同格式的数字 。如 VERIFY 后缀所示，CLTV 如果结果为 FALSE，则停止执行脚本的操作码类型。如果结果为 TRUE，则继续执行。

为了使用 CLTV 锁定输出，将其插入到创建输出的交易中的输出的赎回脚本中。例如，如果 User A 支付 User B 的地址，输出通常会包含一个这样的 P2PKH 脚本：

```
DUP HASH160 <User B's Public Key Hash> EQUALVERIFY CHECKSIG
```

要锁定一段时间，比如说 3 个月以后，交易将是一个 P2SH 交易，其中包含一个赎回脚本：

```
<now + 3 months> CHECKLOCKTIMEVERIFY DROP DUP HASH160 <User B's Public Key Hash> EQUALVERIFY CHECKSIG
```

其中是从交易开始被挖矿时间起计 3 个月的块高度或时间值：当前块高度+12,960（块）或当前 Unix 纪元时间+7,760,000（秒）。之后我们来解释 CHECKLOCKTIMEVERIFY 之后的 DROP 代码：

当 User B 尝试花费这个 UTXO 时，他构建了一个引用 UTXO 作为输入的交易。他使用他的签名和公钥在该输入的解锁脚本，并将交易 nLocktime 设置为等于或更大于 User A 设置的 CHECKLOCKTIMEVERIFY 时间锁。然后，User B 在根源链网络上广播交易。

User B 的交易评估如下。如果 User A 设置的 CHECKLOCKTIMEVERIFY 参数小于或等于支出交易的 nLocktime，脚本执行将继续（就好像执行“无操作”或 NOP 操作码一样）。否则，脚本执行停止，并且该交易被视为无效。 更确切地说，CHECKLOCKTIMEVERIFY 失败并停止执行，标记交易无效（来源：GYLIP-65）：

- 堆栈是空的
- 堆栈中的顶部项小于 0;
- 顶层堆栈项和 nLocktime 字段的锁定时间类型（高度或者时间戳）不相同;
- 顶层堆栈项大于交易的 nLocktime 字段;
- 输入的 nSequence 字段为 0xffffffff。

CLTV 和 nLocktime 使用相同的格式来描述时间锁定，无论是块高度还是自 Unix 纪元以秒钟以来所经过的时间。 最重要的是，在一起使用时，nLocktime 的格式必须与输入中的 CLTV 格式相匹配，它们必须以秒为单位引用块高度或时间。

执行后，如果满足 CLTV，则其之前的时间参数仍然作为堆栈中的顶级项，并且可能需要使用 DROP 进行删除，才能正确执行后续脚本操作码。 为此，您将经常在脚本中看到 CHECKLOCKTIMEVERIFY + DROP 在一起使用。

通过将 nLocktime 与 CLTV 结合使用，交易锁定时间限制中描述的情况发生变化。因为 User A 锁定了 UTXO 本身，所以现在 User B 或 User A 在 3 个月的锁定时间到期之前不可能花费它。

通过将时间锁定功能直接引入到脚本语言中，CLTV 允许在根源链上实现更复杂脚本，从而支持复杂合约。

3.9.2.3 相对时间锁

nLocktime 和 CLTV 都是绝对时间锁定，它们指定绝对时间点。接下来的两个时间锁定功能，我们将要考察的是相对时间锁定，因为它们将消耗输出的条件指定为从块链接中的输出确认起的经过时间。

相对时间锁是有用的，因为它们允许将两个或多个相互依赖的交易链接在一起，同时对依赖于从先前交易的确认所经过的时间的一个交易施加时间约束。换句话说，在

UTXO 被记录在块状块之前，时钟不开始计数。这个功能在双向状态通道和闪电网络中特别有用。

而相对时间锁同时具有交易级功能和脚本级操作码。交易级相对时间锁定是作为对每个交易输入中设置的交易字段 nSequence 的值的共识规则实现的。脚本级相对时间锁定使用 CHECKSEQUENCEVERIFY（CSV）操作码实现。

相对时间锁是在 GYLIP-68 与 GYLIP - 112 的规范基础上来实现的，其中 GYLIP-68 通过与相对时间锁运用一致性增强的数字序列实现，GYLIP-112 中是运用到了 CHECKSEQUENCEVERIFY 这个操作码实现。

3.9.2.4 nSequence 相对时间锁

相对时间锁定可以在每个输入中设置好，其方法是在每个输入中加多一个 nSequence 字段。使用 nSequence 这个字段时，如果输入的交易的序列值小于 2^32（0xFFFFFFFF），就表示尚未“确定”的交易。这样的交易将在内存池中保存，直到被另一个交易消耗相同输入并具有较大 nSequence 值的代替。一旦收到一个交易，其投入的 nSequence 值为 2^32，那么它将被视为“最终确定”。

对于具有 nLocktime 或 CHECKLOCKTIMEVERIFY 的交易，nSequence 值必须设置为小于 2^32，以使时间锁定器有效。通常设置为 2^32 - 1（0xFFFFFFFE）。

由于根源链此处的设计遵循 GYLIP-68，因此新的共识规则适用于任何包含 nSequence 值小于 2^31 的输入的交易（bit 1<<31 is not set）。这意味着如果没有设置最高有效（bit 1<<31），它是一个表示“相对锁定时间”的标志。否则（bit 1<<31set），nSequence 值被保留用于其他用途，例如启用 CHECKLOCKTIMEVERIFY，nLocktime，Opt-In-Replace-By-Fee 等。

一笔输入交易的输入脚本中的 nSequence 值小于 2^31 时，就是相对时间锁定的输入交易。这种交易只有到了相对锁定时间后才生效。例如，具有 30 个区块的 nSequence 相对时间锁的一个输入的交易只有在从输入中引用的 UTXO 开始的时间起至少有 30 个块

时才有效。由于 nSequence 是每个输入字段，因此交易可能包含任何数量的时间锁定输入，所有这些都必须具有足够的时间以使交易有效。

交易可以包括时间锁定输入（nSequence <2^31）和没有相对时间锁定（nsequence> = 2^31）的输入。 nSequence 值以块或秒为单位指定，但与 nLocktime 中使用的格式略有不同。类型标志用于区分计数块和计数时间（以秒为单位）的值。类型标志设置在第 23 个最低有效位（即值 1 << 22）。如果设置了类型标志，则 nSequence 值将被解释为 512 秒的倍数。如果未设置类型标志，则 nSequence 值被解释为块数。

当将 nSequence 解释为相对时间锁定时，只考虑 16 个最低有效位。nSequence 值通常用 16 位掩码（例如 nSequence＆0x0000FFFF）“屏蔽”。

3.9.2.5 带 CSV 的相对时间锁

就像 CLTV 和 nLocktime 一样，有一个脚本操作码用于相对时间锁定，它利用脚本中的 nSequence 值。该操作码是 CHECKSEQUENCEVERIFY，通常简称为 CSV。 在 UTXO 的赎回脚本中评估时，CSV 操作码仅允许在输入 nSequence 值大于或等于 CSV 参数的交易中进行消耗。实质上，这限制了 UTXO 的消耗，直到 UTXO 开采时间过了一定数量的块或秒。

与 CLTV 一样，CSV 中的值必须与相应 nSequence 值中的格式相匹配。如果 CSV 是根据块指定的，那么 nSequence 也是如此。如果以秒为单位指定 CSV，那么 nSequence 也是如此。

当几个（已经形成链）交易被保留为“脱链”时，创建和签名这几个（已经形成链）交易但不传播时，CSV 的相对时间锁特别有用。在父交易已被传播，直到消耗完相对锁定时间，才能使用子交易。

3.9.2.6 中位时间过去 Median-Time-Past

作为激活相对时间锁定的一部分，时间锁定（绝对和相对）的“时间”方式也发生了变化。在根源链中，由于根源链本身是分布式的网络，这意味着每个参与者都有自己的时间观。网络上的事件不会随时随地发生。网络延迟必须考虑到每个节点的角度。最终，所有

内容都被同步，以创建一个共同的分类帐。根源链在过去存在的区块链状态中每个出块周期达成一个新的共识。

区块头中设置的时间戳由矿工设定。共识规则允许一定的误差来解决分散节点之间时钟精度的问题。然而，这诱惑了矿工去说谎，以便通过包括还不在范围内的时间交易来赚取额外矿工费。有关详细信息，请参阅以下部分。

为了杜绝矿工的可能作恶，加强时间安全性，在相对时间锁的基础上根源链同时遵循GYLIP-113，它定义了一个称为“中位时间过去 /（Median-Time-Past）”的新的共识测量机制。通过取最后 11 个块的时间戳并计算其中位数作为“中位时间过去”的值。这个中间时间值就变成了共识时间，并被用于所有的时间计算。过去约两个小时的中间点，任何一个块的时间戳的影响减小了。通过这个方法，没有一个矿工可以利用时间戳从具有尚未成熟的时间段的交易中获取非法矿工费。

Median-Time-Past 更改了 nLocktime，CLTV，nSequence 和 CSV 的时间计算的实现。由 Median-Time-Past 计算的共识时间总是大约在挂钟时间后一个小时。如果创建时间锁交易，那么要在 nLocktime，nSequence，CLTV 和 CSV 中进行编码的估计所需值时，应该考虑它。

3.9.3 根源链闪电网络设计

闪电网络(LN)的重要基础是微支付通道技术。

#1

场景参与者：

- 支付方 User B
- 被支付方 User A

User B 需要 User A 执行某工作，并支付相应酬劳。由于一次性支付总酬劳的信用风险太高，所以酬劳和工作的兑现方式采用小额多次的支付形式。

交易类型

- Funding Transaction，User B 承诺 User A 的务工总酬劳保证金合约交易
- Refund Transaction，User B 防范 User A 抵赖的合约交易
- Commit Transaction，User A 获取务工酬劳的保障合约，亦是防范 User B 抵赖的合约交易
- Settlement Transaction，User A 获取务工酬劳的最终结算合约，属于 Commit Transaction 的最终版本

其中，Funding Transaction 和 Refund Transaction 的 nLockTime 时间需要设定最长。Commit Transaction 会不断的被创建，每次被创建的交易对应的 nLockTime 值逐渐减少。 Settlement Transaction 的 nLockTime=0。这些交易被发送到根源主网后，只有 Settlement Transaction 直接被打包到区块，其余交易因为 nLockTime 时间未消逝殆尽，所以只能呆在内存池中等待。

重点步骤和要素

User B 建立 Funding Transaction，先不发送给 User A 并签名。

User B 建立 Refund Transaction，交易输出指向 User B。User B 发送该交易给 User A，并让 User A 签名，再让 User A 把签好名的交易发回给 User B，User B 持有该交易作为防 User A 抵赖的合约。

User B 再将 Funding Transaction 签名并发送给 User A，告知 User A 对此交易签名并发送到根源链主网。双方都签名(立字为据)，签了名的 Funding Transaction 具备质押性质的合约效力。

务必先让 User A 签名 Refund Transaction 并交回到 User B，然后双方签署 Funding Transaction 并发送到根源主网(微支付通道打开)，避免由于 User A 抵赖，User B 无法解除 Funding Transaction 中锁定的资产。

User B 建立每次酬劳发放的交易 Commit Transaction，交易输入引用 Funding Transaction，交易输出一方指向 User B，一方指向 User A。其中指向 User A 的资产数额作为酬劳。User B 先签名该交易，并发送给 User A。User A 可以选择发送到根源主网，也可以选择不发送。如果 User A 选择发送到根源主网，那么通道两方完成结算，微支付通道关闭。如果 User A 选择不发送，那么务工继续。

每次 User B 创建的 Commit Transaction 都会变更几个交易字段。指向 User B 的输出资产逐渐减少，指向 User A 的输出资产逐渐增多，而且交易的 nLockTime 也在逐渐减少。值得注意的是，由于 Commit Transaction 创建了多次，所以 User A 若将它们都发送到根源主网，那么只有最近一次的 Commit Transaction 会生效，其余 Commit Transaction 会被丢弃。其根本原因在于 nLockTime 值越小就会越优先被打包到区块，如果只关注 nLockTime 一个维度的话，nLockTime=0 对应的交易最快被打包到区块。

最终，Settlement Transaction 实际是 Commit Transaction 的特例，该交易一旦被发送到根源链主网，那么通道两方完成结算，微支付通道关闭。通道关闭的同时，Funding Transaction 解锁，并根据 Settlement Transaction 的输出指向和对应资产份额划拨到 User A 和 User B 的钱包地址。

以上步骤最终实现为类似于 LN 的根源链上层应用业务功能。最终的端用户并不会感知这些交互和数据结构的变化。

以上设计仍有缺陷，例如：通道的构建是单向的，如何构建双向的通道呢？如果通道两方任何一方抵赖，那么剩下的一方发送手里的防抵赖合约交易到根源链主网就能够避免损失，但是需要等待 nLockTime 的消逝时间窗口。如何避免等待，立即止损，并且相应的给出抵赖一方惩罚？如果通道多方连接形式为 A-B-C-D，那么 A 需要和 C 或者 D 交易，如何跨节点支付？解决方案是：RMSC 与 HTLC 技术

3.9.4 RMSC

闪电网络的基础是交易双方之间的双向微支付通道，RSMC（Recoverable Sequence Maturity Contract）定义了该双向微支付通道的最基本工作方式。

微支付通道中沉淀了一部分资金，也记录有双方对资金的分配方案。通道刚设立时，初始值可能是{User A: 0.4, User B: 0.6}，这意味着打入通道的资金共有 1.0 BSTK，其中 User A 拥有 0.4 BSTK，User B 拥有 0.6 BSTK。通道的设立会记录在根源链区块链上。

假设稍后 User B 决定向 User A 支付 0.1 BSTK。双方在链下对最新余额分配方案{User A: 0.5, User B: 0.5}进行签字认可，并签字同意作废前一版本的余额分配方案{User A: 0.4, User B: 0.6}，User A 实际上就获得了 0.5 BSTK 的控制权。如图所示。

如果 User A 暂时不需要将通道中现在属于她的 0.5 BSTK 用作支付，那么她可以无须及时更新区块链上记录的通道余额分配方案，因为很可能一分钟后 User A 又需要反过来向 User B 支付 0.1 BSTK，此时他们仍然只需要在链下对新的余额分配方案达成一致，并设法作废前一版本的余额分配方案就行了。

类型	冻结	User A	User B
无条件	1.0	0.4	0.6

类型	冻结	User A	User B
无条件	1.0	0.5	0.5

如果 User A 打算终止通道并动用她的那份资金，那么她可以向区块链出示双方签字的余额分配方案。如果一段时间之内 User B 不提出异议，那么区块链会终止通道并将资金按协议转入各自预先设立的提现地址。如果 User B 能在这段时间内提交证据证明 User A 企图使用的是一个双方已同意作废的余额分配方案，则 User A 的资金将被罚并给到 User B。

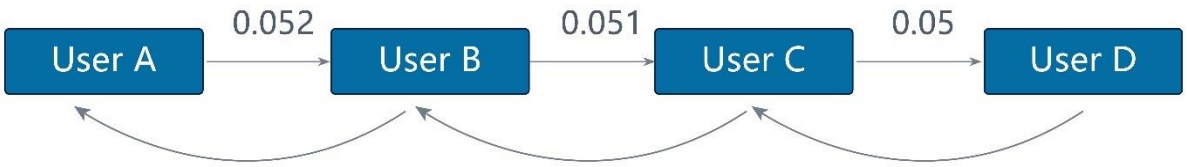
由于理论过程过于繁杂，而且本文主要目的是闪电网络相关概念的串联和导引，所以细节不赘述。

3.9.5 HTLC

RSMC 只支持最简单的无条件资金支付，HTLC（Hashed Timelock Contract）则进一步实现了有条件的资金支付，通道余额的分配方式也因此变得更为复杂。

通过 HTLC，User A 和 User B 可以达成这样一个协议：协议将锁定 User A 的 0.1 BSTK，在时刻 T 到来之前（T 以未来的某个区块链的高度来表述），如果 User B 能够向 User A 出示一个适当的 R（称为秘密），使得 R 的 Hash 值等于事先约定的值 H(R)，User B 就能获得这 0.1 BSTK；如果直到时刻 T 过去 User B 仍然未能提供一个正确的 R，那么这 0.1 BSTK 将自动解冻并归还 User A。

由于到期时间 T、提款条件 H(R)、支付金额和支付方向的不同，同一个通道上可以同时存在多个活动的 HTLC 合约，再加上唯一的通过 RSMC 商定的无条件资金余额，余额分配方式会变得相当复杂。



User A 想给 User D 发送 0.05 BSTK，但 User A 和 User D 之间并没有微支付通道。User A 找到了一条经过 User B、User C 到达 User D 的支付路径，该路径由 User A/User B、User B/User C 和 User C/User D 这样三个微支付通道串接而成。

此时，User D 生成了一个秘密 R 并将 Hash(R)发送给 User A，User A 不需要知道 R。该交易的流程具体如下。

1. User A 和 User B 商定一个 HTLC 合约：只要 User B 能在“3”天内向 User A 出示 Hash 正确的 R，User A 就会支付 User B 0.052 BSTK；如果 User B 做不到，这笔钱将 3 天后自动退还 User A。

2. 同样地，User B 和 User C 商定一个 HTLC 合约：只要 User C 能在“2”天内向 User B 出示 Hash 正确的 R，User B 就会支付 User C 0.051 BSTK；如果 User C 做不到，这笔钱到期后将自动退还 User B。

3. 最后，User C 和 User D 商定一个 HTLC 合约：只要 User D 能在“1”天内向 User C 出示 Hash 正确的 R，User C 就会支付 User D 0.05 BSTK；如果 User D 做不到，这笔钱到期后将自动退还 User C。

3.10 智能合约

根源链 BOS 的合约层技术实现符合可插拔要求，面向合约技术而言，当前支持 RSK、MAST、LN、CounterParty 等主流合约技术，并基于其上来实现根源链自有的 SC-LN 等技术。

Rootstock（RSK）是基于侧链技术实现，其目标是通过实现智能合约、即时支付和更高的可扩展性等问题，为主链赋能，实现主链的智能合约功能，为主链生态系统增加价值和实用性。关于侧链技术已经阐释不再赘述。

引入 RSK 作为根源链 BOS 的合约层技术实现之一，是基于：

(1) 主链无智能合约功能或者脚本系统功能较为简单

(2) 主链确权速度慢

(3) 对等网络速度慢

(4) 网络拥堵

(5) 交易手续费高

具体实现为：

(1) 引入智能合约的图灵完备性的虚拟机（RVM）

(2) 门限签名方案实现的安全联合工作量证明挖矿机制

(3) 嵌入延迟低的中继骨干网络支撑点对点通讯

(4) 2WPP

MAST 方案由三部分构成，分别是支付给脚本 hash(P2SH)，抽象语法树 AST 和 Merkle 树。MAST 的典型实现是基于脚本系统，AST 给出了脚本语法的分解规则。根源链 BOS 基于 MAST 扩充了脚本系统的语法分析树，提高了语法分析的效率。

根源链 BOS 目前正在尝试引入 XCP 的实现，用于促使智能合约的多样性。因为 XCP 具备“燃烧”的概念，它符合根源链 BSTK 销毁机制的需要。

根源链 BOS 引入了闪电网络 LN 的基础技术并加以改进，用以形成 SC-LN。 这是因为 LN 不仅具备分布式系统的特征，而且具备良好的独立性，易于插拔。基于 SC-LN，无需信任对方以及第三方即可实现实时海量的交易网络。LN 是基于微支付通道演进而来，创造性的设计出了两种类型的交易合约：序列到期可撤销合约 RSMC（Revocable Sequence Maturity Contract，哈希时间锁定合约 HTLC（Hashed Timelock Contract）。基于以上合约技术的稳定与高效，RSMC 和 HTLC 合约技术也同样作为根源链 BOS 合约层的实现技术基础来使用。

3.10.1 合约模板

合约模板处于应用服务层，面向 SDK 和网关 API 进行了封装层面的选择设计。首先允许使用方选择合约类型，其次提供合约模板接口。

3.10.2 编译器

当前合约层具备基于 RSK、MAST 协议实现的编译器 SCRSK compiler、SCMAST compiler。

3.10.3 解释器

当前合约层具备基于 RSK、MAST 协议实现的 VM 包括 SCRVM、SCMVM。

3.10.4 根源链智能合约设计

虽然区块链合约（下称合约）当前发展迅速而且种类繁多，但是主要分为 4 大类：根源链类，侧链类，以太坊类，混合类。其中混合类诸如 EOS 的 WASM（WebAssembly）、SIA 的文件合约（File Contract）、IPFS 结合以太坊合约（Eth Contract）、BSTK 结合以太坊合约（Eth Contract）等等。

接下来的篇幅主要是集中在 BSTK 的合约设计与实现方面。BSTK 属于混合类合约实现方案，结合了根源链、侧链、以太坊、交易压缩、闪电网络、隔离见证等若干技术。

智能合约（Smart contract）是一种旨在以信息化方式传播、验证或执行合同的计算机协议。智能合约允许在没有第三方的情况下进行可信交易。这些交易可追踪且不可逆转。智能合约概念于 1994 年由 Nick Szabo 首次提出。技术角度：

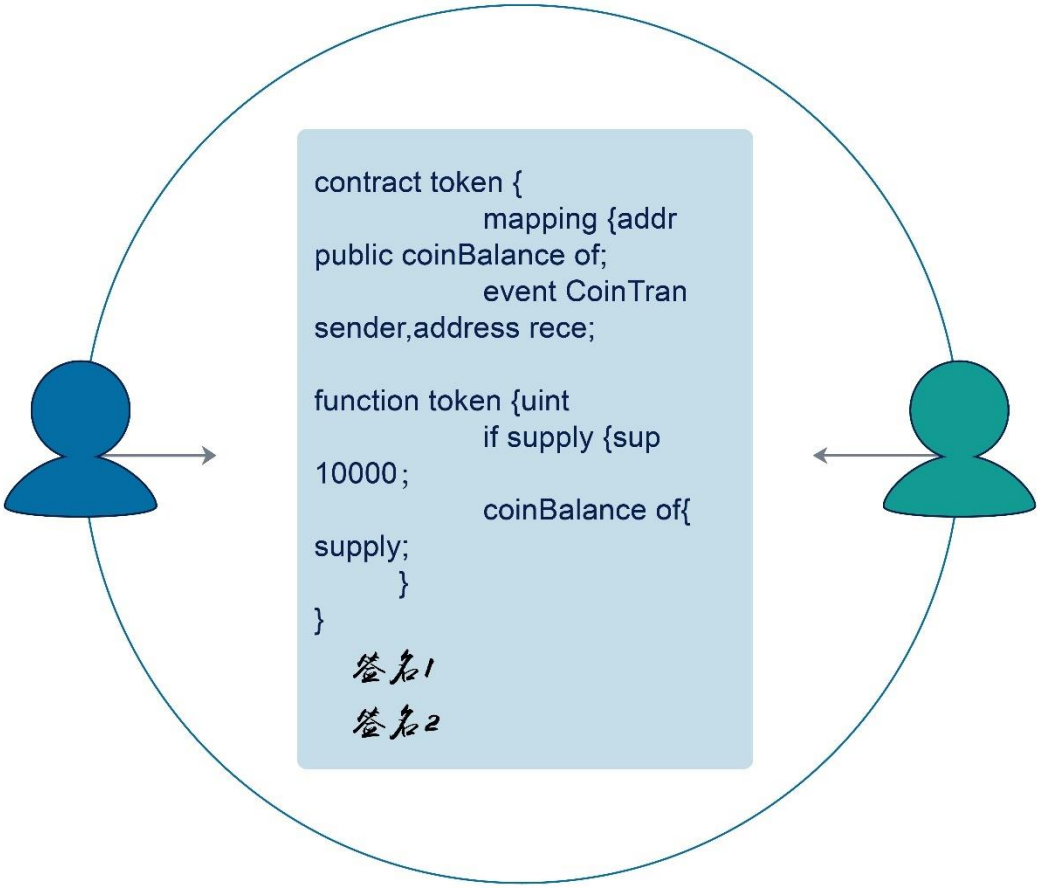
[1] 合约参与方编写代码并签名，然后上链（托管第三方）

[2] 链上验证、共识、履行(执行)合约

[3] 履行成功，参与方资产按照合约约定重新分配；履行失败，参与方资产将原路返回

平台角度：类 DAO（Decentralized Autonomous Organization），类众筹平台（Crowdfunding Platform）。传统众筹平台需要合约第三方参与，属于典型的中心化方

案，缺点有很多，不做赘述。智能合约平台也需要合约第三方参与，不过使用的是区块链方案，借助区块链的分布式计算、共识、密码学等优势来达成合约的审计与履行。



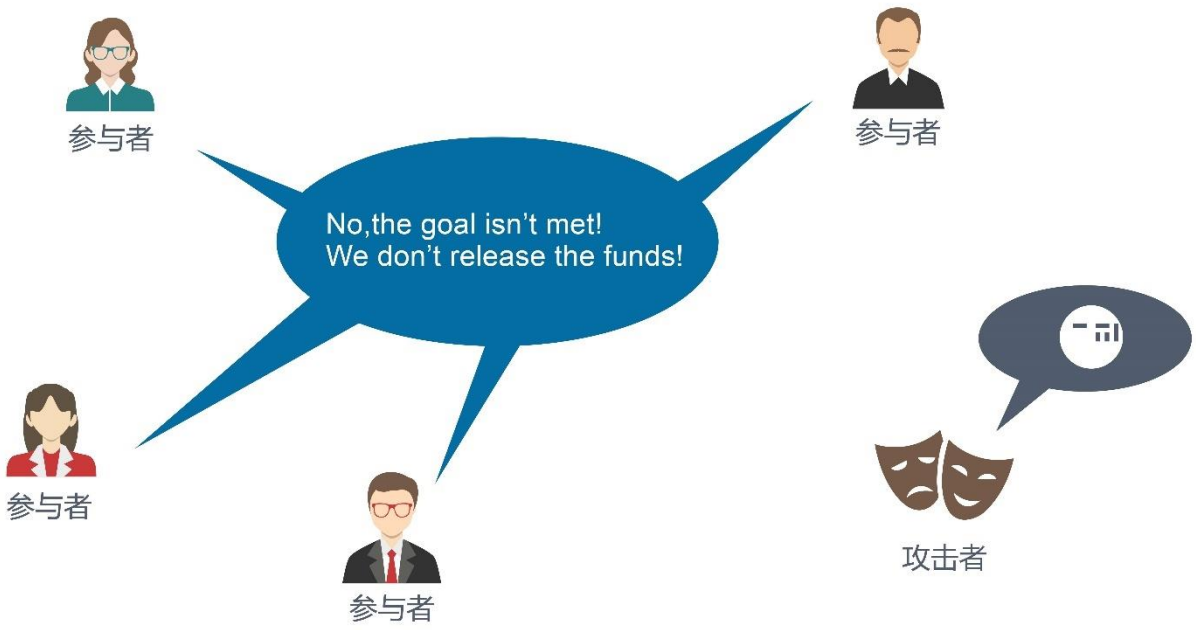
所有合约都是基于区块链技术之上来阐述，所以智能合约优势的发挥离不开区块链的属性。

3.10.5 古典合约痛点

没有区块链，合约以及履行合约的系统只能依靠本身就是不可靠的中心化第三方。制定和履行合约的人会犯错，会带来合约所描述的业务风险以及实施代价增高；合约数据易被篡改或灭失。使用区块链，合约以及履行合约的系统则会具有下列优势。

[1] 区块链不可变性，这意味着智能合约永远不会改变，没有人可以篡改或违反合约

[2] 分布式，这意味着合约的结果由网络中的每个人（网络节点）验证，就像区块链上的任何交易一样。分布式使攻击者无法强制控制释放权益归属资产，因为所有其他参与者都会检测到此类尝试并将其标记为无效（单点攻击无效）。



3.10.6 根源链智能合约分析

智能合约当前仍旧出于萌芽期。以太坊的智能合约漏洞百出，而根源链所基于的类 bitcoin 系统依旧是最稳定和难以攻破的系统，所以本文使用 BSTK 方案作为智能合约设计与实现的技术选择。关于为什么以太坊智能合约漏洞百出，可参见 MIT 技术洞见。

智能合约不具备灵活性。相比于传统合约或者法律凭证，如果需要调整，那么得到的只能是干巴巴的拒绝，这会导致智能合约技术的使用范畴变的狭窄。

智能合约的双刃剑。当智能合约设计为图灵完备时，将导致合约的复杂度无限升高。复杂意味着更多能力，但也伴随着更多错误以及惨痛的代价。脚本系统是非图灵完备的，这也是为什么根源链系统非常牢靠的原因之一。

智能合约是合约吗？不是，确切的说是“代码约定”，因为智能合约并不具备符合人类社会一般要求的合约属性。当下大多数情况，所谓“智能合约”经常沦为利益驱使炒作的技术噱头。

3.10.7 根源链智能合约优势

- 直接与对方打交道。 智能合约消除了对中间方的需求，并允许与对方建立透明，直接的关系。
- 分布、可用、严格履行。 由于参与者不依赖于第三方，因此没有任何一个人或实体控制数据或资产。 即使任何个人离开区块链网络，网络也将继续运行而不会丢失数据或失去完整性。
- 信任。 区块链协议决定数据一经登记上链是不可改变的，智能合约同样作为上链数据具备不可变性。
- 欺诈减少。 由于智能合约存储在分布式区块链网络中，因此其结果将由该网络中的每个人进行验证。 因此，没有人可以强制控制释放其他人的资金或数据，因为所有其他区块链参与者都会发现这一点并将此类尝试标记为无效。

- 成本效益。 消除中间方可以免除额外费用，使企业及其客户不仅可以直接进行互动和交易，而且可以在交易中收取低费用或不收取任何费用。
- 持续记录与审计。 所有合约、交易都按时间顺序存储在区块链中，并且通过区块链可追溯性达到审计合约履行结果的目的。

3.10.8 根源链智能合约应用范畴

智能合约相对应用范围比较广泛，在很多业务领域都能够见到智能合约的身影。常见的应用领域例如物联网、分布式电商、分布式存储、医疗大数据、人工智能等。值的思考的是随着业务领域的复杂度升高，合约的拟定与履行将成为一件复杂的事情。如果停留在普通货币资产层面，合约的实施比较现实和可行，但是进入到自治层面，例如自治业务、自治组织、自治政府、自治社会，合约的实施还有待观察，毕竟越倾向于社会层面，其维度与复杂度都是难以想象的。

3.10.9 根源链多重签名技术

基本的根源链地址以数字 1 开头，来源于公钥，而公钥来源于私钥。虽然任何人都可以将根源链发送到 一个 1 开头的地址，但根源链只能在通过相应的私钥签名和公钥哈希值后才能消费。

以数字 3 开头的根源链地址是 P2SH 地址，有时被错误的称谓多重签名或多重签名地址。他们指定根源链交易中受益人为哈希的脚本，而不是公钥的所有者。这个特性基于 GYLIP0016 引进，目前因为 GYLIP0016 提供了增加功能到地址本身的机会而被广泛的采纳。不同于 P2PKH 交易发送资金到传统 1 开头的根源链地址，资金被发送到 3 开头的地址时，需要的不仅仅是一个公钥的哈希值和一个私钥签名作为所有者证明。在创建地址的时候，这些要求会被指定在脚本中，所有对地址的输入都会被这些要求阻隔。

一个 P2SH 地址从交易脚本中创建，它定义谁能消耗这个交易输出（后面“P2SH（Pay-to-Script-Hash）”一节对此有 详细的介绍）。编码一个 P2SH 地址涉及使用一个在创建根源链地址用到过的双重哈希函数，并且只能应用在脚本而不是公钥：

```
script hash = RIPEMD160(SHA256(script))
```

产生的脚本哈希由 Base58Check 编码前缀为 5 的版本、编码后得到开头为 3 的编码地址。一个 P2SH 地址例子是 3F6i6kwkevjR7AsAd4te2YB2zZyASEm1HM。可以使用根源链 Explorer 命令脚本编码获得，比如 sha256, ripemd160, and base58check-encode，举例如下：

```
$ echo dup hash160 [ 89abcdefabbaabbaabbaabbaabbaabbaabbaabba ] equalverify checksig > script

$ bx script-encode < script | bx sha256 | bx ripemd160 | bx base58check-encode --version 5

3F6i6kwkevjR7AsAd4te2YB2zZyASEm1HM
```

P2SH 不一定是多重签名的交易。虽然 P2SH 地址通常都是代表多重签名，但也可能是编码其他类型的交易脚本。

3.10.10 多签地址与 P2SH

目前，P2SH 函数最常见的实现是多重签名地址脚本。顾名思义，底层脚本需要多个签名来证明所有权，此后才能消费资金。设计根源链多重签名特性是需要从总共 N 个密钥中需要 M 个签名（也被称为“阈值”），被称为 M-N 多签名，其 中 M 是等于或小于 N。例如，第一章中提到的咖啡店主 User B 使用多重签名地址需要 1-2 签名，一个是属于他的密钥和一个属于他同伴的密钥，以确保其中一方可以签署消费一笔锁定到这个地址的输出。这类似于传统的银行中的一个“联合账户”，其中任何一方配偶可以单独签单消费。或就像 User B 雇佣的网页设计师， 创立一个网站，可能为他的业务需要一个 2-3 的多签名地址，确保除非至少两个业务合作伙伴签署签名交易才可以进行支付消费。

3.10.10.1 多重签名

多重签名脚本设置了一个条件，其中 N 个公钥被记录在脚本中，并且至少有 M 个必须提供签名来解锁资金。这也称为 M-N 方案，其中 N 是密钥的总数，M 是验证所需的签名的数量。例如，2/3 的多重签名是三个公钥被列为潜在签名人，至少有 2 个有效的签名才能花费资金。此时，标准多重签名脚本限制在最多 15 个列出的公钥，这意味着您可以从 1 到 15 之间的多重签名或该范围内的任何组合执行任何操作。之前，限制 15 个已列出 d 的密钥可能会被解除，因此请检查 isStandard（）函数以查看当前网络接受的内容。

设置 M-N 多重签名条件的锁定脚本的一般形式是：

```
M <Public Key 1> <Public Key 2> ... <Public Key N> N CHECKMULTISIG
```

M 是花费输出所需的签名的数量，N 是列出的公钥的总数。 设置 2 到 3 多重签名条件的锁定脚本如下所示：

```
2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG
```

上述锁定脚本可由含有签名和公钥的脚本予以解锁： 或者由 3 个存档公钥中的任意 2 个相一致的私钥签名组合予以解锁。 两个脚本组合将形成一个验证脚本：

<Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG

当执行时，只有当未解锁版脚本与解锁脚本设置条件相匹配时，组合脚本才显示得到结果为真（Ture）。

上述例子中相应的设置条件即为：未解锁脚本是否含有 3 个公钥中的任意 2 个相对应的私钥的有效签名。

CHECKMULTISIG 的执行中有一个 bug，需要一些轻微的解决方法。 当 CHECKMULTISIG 执行时，它应该消耗在堆栈上的 M + N + 2 个项目作为参数。 然而，由于该错误，CHECKMULTISIG 将弹出（pop）超出预期的额外值或一个值。 我们来看看这个更详细的/使用以前的/验证示例：

<Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG

首先，CHECKMULTISIG 弹出最上面的项目，这是 N（在这个例子中 N 是“3”）。然后它弹出 N 个项目，这是可以签名的公钥。在这个例子中，公钥 A，B 和 C.然后，它弹出一个项目，即 M，仲裁（需要多少个签名）。这里 M = 2。此时，CHECKMULTISIG 应弹出最终的 M 个项目，这些是签名，并查看它们是否有效。

然而实施中的错误导致 CHECKMULTISIG 再弹出一个项目（总共 M + 1 个）。检查签名时，不考虑额外的项目，因此它对 CHECKMULTISIG 本身没有直接影响。但是，必须存在额外的值，因为如果不存在，则当 CHECKMULTISIG 尝试弹出空堆栈时，会导致堆栈错误和脚本失败（将交易标记为无效）。因为额外的项目被忽略，它可以是任何东西，但通常使用 0。

因为这个 bug 成为共识规则的一部分，所以现在它必须永远被复制。因此，正确的脚本验证将如下所示：

0 <Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3 CHECKMULTISIG

这样解锁脚本就不是下面的：

<Signature B> <Signature C>

而是：

0 <Signature B> <Signature C>

从现在开始，如果你看到一个 multisig 解锁脚本，你应该期望看到一个额外的 0 开始，其唯一的目的是解决一个 bug，结果现在已经成为一个共识规则的解决方法。

3.10.10.2 P2SH

P2SH 作为一种强大且能大大简化复杂交易脚本的交易类型而引入。为进一步解释 P2SH 的必要性，让我们先看一个实际的例子。

举例介绍 Market A，一个根源链参与者，并假定他使用根源链通卡来实现支付结算。Market A 的公司采用根源链多重签名作为其公司会计账簿记账要求。多重签名脚本是根源链高级脚本最为常见的运用之一，是一种严谨的脚本。针对所有的顾客支付（即应收账款），Market A 的公司要求采用多重签名交易。基于多重签名机制，顾客的任何支付都需要至少两个签名才能解锁，一个来自 Market A，另一个来自其合伙人或拥有备份钥匙的代理人。这样的多重签名机制能为公司治理提供管控便利，同时也能有效防范盗窃、挪用和遗失。 最终的脚本非常长：

2 <Market A's Public Key> <Partner1 Public Key> <Partner2 Public Key> <Partner3 Public Key> <Attorney Public Key> 5 OP_CHECKMULTISIG

虽然多重签名十分强大，但其使用起来还是多有不便。基于之前的脚本，Market A 必须在客户付款前将该脚本发送给每一位客户，而每一位顾客也必须使用特制的能产生客户交易脚本的根源链钱包软件，每位顾客还得学会如何利用脚本来完成交易。

此外，由于脚本可能包含特别长的公钥，最终的交易脚本可能是最初交易脚本长度的 5 倍之多。额外长度的脚本将给客户造成费用负担。最后，一个长的交易脚本将一直记录在所有节点的随机存储器的 UTXO 集中，直到该笔资金被使用。采用这种复杂输出脚本使得在实际交易中变得困难重重。

P2SH 正是为了解决这一实际难题而被引入的，它旨在使复杂脚本的运用能与直接向根源链地址支付一样简单。在 P2SH 支付中，复杂的锁定脚本被电子指纹所取代，电子指纹是指密码学中的哈希值。

当一笔交易试图支付 UTXO 时，要解锁支付脚本，它必须含有与哈希相匹配的脚本。P2SH 的含义是，向与该哈希匹配的脚本支付，当输出被支付时，该脚本将在后续呈现。

在 P2SH 交易中，锁定脚本由哈希运算后的 20 字节的散列值取代，被称为赎回脚本。因为它在系统中是在赎回时出现而不是以锁定脚本模式出现。

Locking Script	2 Pubkey 1 Pubkey2 Pubkey3 Pubkey4 Pubkey5 5 CHECKMULTISIG
Unlocking Script	Sig 1 Sig 2

Redeem Script	2 Pubkey 1 Pubkey2 Pubkey3 Pubkey4 Pubkey5 5 CHECKMULTISIG
Locking Script	HASH160 <20-byte hash of redeem script> EQUAL
Unlocking Script	Sig1 Sig2 <redeem script>

从表中可以看出，对于 P2SH，详细描述了输出（赎回脚本）的条件的复杂脚本不会在锁定脚本中显示。

相反，只有它的散列值在锁定脚本中呈现，并且兑换脚本本身稍后呈现，作为解锁脚本在输出花费时的一部分。 这使得给矿工的交易费用从发送方转移到收款方，复杂的计算工作也从从发送方转移到收款方。

输出脚本（Redeem Script）中的**“2 Pubkey1 Pubkey2 Pubkey3 Pubkey4 Pubkey5 5 CHECKMULTISIG”**的内容，没有出现在锁定脚本（Locking Script 表中第二行内容）中，但对实现上很长的一大串的**“2 Pubkey1 Pubkey2 Pubkey3 Pubkey4 Pubkey5 5 CHECKMULTISIG”**（有 520 字节）进行哈希运算后的 20 字节的散列值取代之，然后将之（**“2 Pubkey1 Pubkey2 Pubkey3 Pubkey4 Pubkey5 5 CHECKMULTISIG”**）放到解锁脚本中（Unlocking Script）。这使得给矿工的交易费用从发送方转移到收款方，并且令复杂的计算工作也从从发送方转移到收款方。

让我们再看下 Market A 公司的例子，复杂的多重签名脚本和相应的 P2SH 脚本。 首先，Market A 公司对所有顾客订单采用多重签名脚本： 2 <Market A's Public Key> 5 CHECKMULTISIG 如果占位符由实际的公钥（以 04 开头的 520 字节）替代

整个脚本都可由仅为 20 个字节的密码哈希所取代，首先采用 SH256 哈希算法，随后对其运用 RIPEMD160 算法。20 字节 的脚本为：

```
54c557e07dde5bb6cb791c7a540e0a4796f5e97
```

一笔 P2SH 交易运用锁定脚本将输出与哈希关联，而不是与前面特别长的脚本所关联。使用的锁定脚本为：

```
HASH160 54c557e07dde5bb6cb791c7a540e0a4796f5e97e EQUAL
```

这样处理后，脚本的长度有效缩短。取代“向该 5 个多重签名脚本支付”，这个 P2SH 等同于“向含该哈希的脚本支付”。顾客在向 Market A 公司支付时，只需在其支付指令中

纳入这个非常简短的锁定脚本即可。当 Market A 想要花费这笔 UTXO 时，附上原始赎回脚本（与 UTXO 锁定的哈希）和必要的解锁签名即可，如：

```
<Sig1> <Sig2> <2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG>
```

两个脚本经由两步实现组合。 首先，将赎回脚本与锁定脚本比对以确认其与哈希是否匹配：

```
<2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG> HASH160 <redeem scriptHash> EQUAL
```

假如赎回脚本与哈希匹配，解锁脚本会被执行以释放赎回脚本：

```
<Sig1> <Sig2> 2 PK1 PK2 PK3 PK4 PK5 5 CHECKMULTISIG
```

P2SH 的另一重要特征是它能将脚本哈希编译为一个地址（其定义请见 GYLIP0013/GYLIP-13）。P2SH 地址是基于 Base58 编码的一 个含有 20 个字节哈希的脚本，就像根源链地址是基于 Base58 编码的一个含有 20 个字节的公钥。由于 P2SH 地址采用 5 作为前缀，这导致基于 Base58 编码的地址以“3”开头。例如，Market A 的脚本，基于 Base58 编码下的 P2SH 地址变 为“39RF6JqABiHdYHkfChV6USGMe6Nsr66Gzw”。

此时，Market A 可以将该地址发送给他的客户，这些客户可以 采用任何的根源链钱包实现简单支付，就像这是一个根源链地址一样。以“3”为前缀给予客户这是一种特殊类型的地址的暗示，该地址与一个脚本相对应而非与一个公钥相对应，但是它的效果与根源链地址支付别无二致。 P2SH 地址隐藏了所有的复杂性，因此，运用其进行支付的人将不会看到脚本。

3.10.10.3 优势

与直接使用复杂脚本以锁定输出的方式相比，P2SH 具有以下特点：

- 在交易输出中，复杂脚本由简短电子指纹取代，使得交易代码变短。

- 脚本能被编译为地址，支付指令的发出者和支付者的根源链钱包不需要复杂工序就可以执行 P2SH。
- P2SH 将构建脚本的重担转移至接收方，而非发送方。
- P2SH 将长脚本数据存储的负担从输出方（存储于 UTXO 集，影响内存）转移至输入方（存储在区块链里面）。
- P2SH 将长脚本数据存储的重担从当前（支付时）转移至未来（花费时）。
- P2SH 将长脚本的交易费成本从发送方转移至接收方，接收方在使用该笔资金时必须含有赎回脚本。

3.10.10.4 赎回脚本和标准确认

最初 P2SH 仅限于标准根源链交易脚本类型（即通过标准函数检验的脚本）。这也意味着使用该笔资金的交易中的赎回脚本只能是标准化的 P2PK、P2PKH 或者多重签名，而非 RETURN 和 P2SH。

根源链核心客户端中，P2SH 交易能包含任意有效的脚本，这使得 P2SH 标准更为灵活，也可以用于多种新的或复杂类型的交易。

请记住不能将 P2SH 植入 P2SH 赎回脚本，因为 P2SH 不能自循环。虽然在技术上可以将 RETURN 包含在赎回脚本中，但由于规则中没有策略阻止您执行此操作，因此在验证期间执行 RETURN 将导致交易被标记为无效，因此这是不实际的。

需要注意的是，因为赎回脚本只有在你试图发送一个 P2SH 输出时才会在根源链网络中出现，假如你将输出与一个无效的交易哈希锁定，则它将会被忽略。该 UTXO 将会被成功锁定，但是你将不能使用该笔资金，因为交易中含有赎回脚本，该脚本因是一个无效的脚本而不能被接受。这样的处理机制也衍生出一个风险，你可能将根源链锁定在一个未来不能被花费的 P2SH 中。因为根源链网络本身会接受这一 P2SH，即便它与无效的赎回脚本所对应（因为该赎回脚本哈希没有对其所表征的脚本给出指令）。

P2SH 锁定脚本包含一个赎回脚本 HASH，该脚本对于赎回脚本本身未提供任何描述。P2SH 交易即便在赎回脚本无效的情况下也会被认为有效。如果处理不当，有可能会出现一个事故，即你的根源链可能会被锁死在 P2SH 这个交易中，导致你以后再也不能花费这笔根源链了。

3.11 账户密钥

账户密钥层主要包含两部分，第一是已经封装的分布式存储依赖的账户密钥组件 KMS 和 Barbican 暴露的接口，第二是根源链 BOS 分布式账户密钥管理系统。

根源链 BOS 分布式账户密钥管理系统是基于 NuCypher 实现，SCNC KMS(分布式去中心化的密钥管理服务)可以帮助 DApp 开发者对其分布式代理进行重新加密作为服务的区块链上的数据进行安全保护。关于 DApp 开发的 SDK 和网关 API 已经将这一部分重新封装。

根源链 BOS 账户密钥层的系统 SCNC KMS 具备账户和密钥管理两部分，允许用户把密文从一个公钥转换到另外一个公钥中，使用其再加密钥，用户不需要了解任何的基础信息。其核心的基础设施就是能够使开发者得以储存、共享和管理公共区块链上的私人数据。

3.11.1 账户接口

使用者通过用户服务层系统管理的 DashBoard 或者使用应用服务层命令行工具，轻松创建、导入和轮换密钥，以及定义使用策略和审计使用情况。

3.11.2 账户密钥

SCNC KMS 为您提供加密密钥和账户的控制。SCNC KMS 当前提供的能力包括如下。

- 主密钥（无论是导入的还是由 KMS 自行创建的）都以加密格式存储在高持久性的存储中，以帮助确保在需要时可对其进行检索。
- 每年自动轮换一次在 KMS 中创建的主密钥，而无需重新加密已使用主密钥加密过的数据。
- 无需记录旧版主密钥，因为 KMS 会保持其可用，以解密以前加密的数据。
- 创建新的主密钥，并对谁有权访问这些密钥以及它们可用于哪些服务随时加以控制。
- 从自己的密钥管理基础设施导入密钥并在 KMS 中使用。

3.12 系统管理

系统管理主要包括管理接口、配置、监控、分析仪表。其中管理接口是负责调度配置接入，便于运维人员进行操作；通过管理接口调度监控服务，便于系统管理员实时查看平台的运行情况；通过管理接口调度分析仪表服务，方便使用者查看并允许操作响应模块，例如账户密钥。

3.12.1 管理接口

管理接口的实现基于 Cockpit。

3.12.2 配置

配置模块分为两部分，一部分是 GUI，另一部分是 CLI。

3.12.3 监控

监控模块的实现基于 Prometheus。

3.12.4 分析仪表

分析仪表的实现基于 Grafana。

3.13 SDK

SDK 协议层的实现基于 IDL 规范，由于面向开发者和面向基础服务层两部分，所以根源链 BOS 在考虑实现 SDK 时，不仅要符合低耦合、易扩展、兼容等要求，还要具备向后兼容便于升级更新。

3.13.1 协议层描述语言 SCIDL/PDL

SDK 协议层实现使用 SCIDL/PDL，定义了接口和精简分布式对象的过程。作为一种类似 Java 的规范语言，SCIDL/PDL 用于分离对象的接口与其实现，且剥离了编程语言和硬件的依赖性。SCIDL/PDL 使用 IDL/PDL 定义接口的客户端，而应用端的开发者并不知道接口背后的实现细节。SCIDL/PDL 基于此可提供一套通用的数据类型，并以这些数据类型来定义更为复杂的数据类型。

3.13.2 SDK 实现设计

由于 SDK 的特殊性，根源链 BOS 在 SDK 实现层一开始对于 SDK 的一些通用的整体的元素的设计做了充分和反复的考量。因为 SDK（尤其平台 SDK,使用的应用成百上千）一个及其细微的调整都会影响很多开发者的版本周期。因此前期的设计显得尤为重要。

当前根源链 BOS 实现 SDK 主要支持 C++、Java、Go、Python，由于不同语言的差异性，叙述较为复杂，所以这里仅描述我们在实现时的原则。

- 接口名称、参数名称要足够清晰
- 一个接口只做一件事
- 接口参数要尽可能少
- 接口参数要一定要校验、需要转义或者转换的一定要尽可能早的处理
- 通用的名称要统一
- 同步和异步接口
- 多线程
- 第三方平台
- 配置

3.14 工具与服务

根源链 BOS 在应用服务层提供的工具包括协议层查看器、分布式路由工具集、分布式转发工具集、分布式存储工具集、分布式加密工具集、合约交互命令行工具(包括编译器)、账户密钥工具集、系统管理接口调度器(命令行方式)、侧链适配器工具集、区块链层工具集(包括交易、钱包驱动)

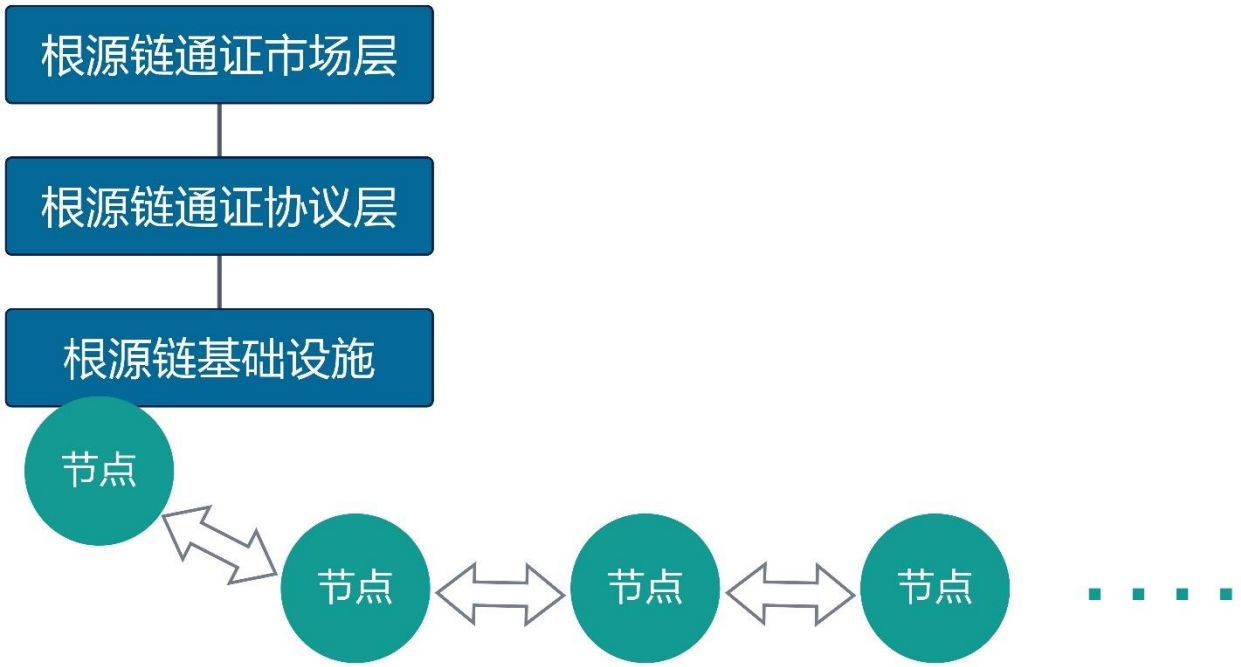
根源链 BOS 在应用服务层提供的服务包括区块链层后台服务、分布式路由服务、分布式转发服务、分布式存储服务簇、分布式加密服务、合约解释器服务、分布式账户密钥服务、系统管理服务簇、侧链协议服务、网管 API 后台服务

3.15 网关 API

URL 使用 RESTful 风格，通讯数据使用 Json 与 ProtoBuf 风格。

3.16 根源链通证协议层与虫洞协议

3.16.1 架构

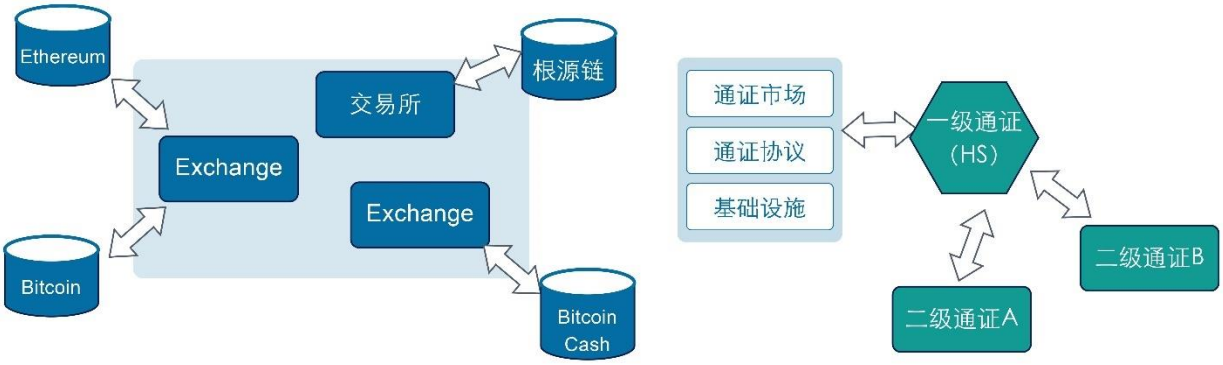


根源链是一个基于数据信息追溯及确权的泛物联网应用公有链基础设施，以根源卡 BSTK 为资产交易流通的凭证。

根源链通证协议层（Token Protocol Layer，TPL）是建立在根源链基础设施之上的附生层，通过特定比例将根源卡与哈耶克种子（Hayek Seed，HS）进行兑换。市场参与者可基于 HS 自主构建符合自己需要的、可自定义类型的资产或通证及其数量。

根源链通证市场层（Token Market Layer，TML）是建立在根源链通证协议层之上，各市场通过自定义类型的资产进行符合自己的业态设计与开发。例如钱包、资产发布、交易所，以及行业应用等。

3.16.2 业务模型



首先，根源链基础设施以根源卡为凭证，通过特定比例将根源卡与哈耶克种子（Hayek Seed，HS）进行兑换。其中哈耶克种子作为一级通证。

其次，在通证协议层，市场参与者能够自主构建自定义类型通证，例如二级通证 A。通过消耗特定 HS，从而构建出满足自己业态的二级通证。例如，类型为书法类的字通证，书法类的画通证，书法类的雕刻品通证等。

第三，建立在根源链基础设施与交易所之上的通证市场，各参与者能够轻松地实现自有通证与 HS 的兑换，以及与根源卡、比特币、以太坊、比特现金进行兑换。

3.16.3 根源链通证协议层

3.16.3.1 定义

根源链通证协议层是一个基于根源链的开源分布式资产平台，通证协议层（TPL）是一个创建和交易自定义数字资产和货币的平台。 TPL 交易是根源链交易，可在根源链区块链上实现附生功能。 根源链是一个共有链，目前基于根源链核心客户端实现通证协议层，提供根源链的所有功能以及先进的 TPL 功能。

使用 TPL，创建通证以表示自定义货币或资产并通过根源链区块链进行交易非常简单。 TPL 提供的强大功能和简单性使其成为领先的基于根源链的通证协议。

TPL 可以轻松实现分布式众筹或类似功能。参与者可以将根源卡直接发送到发行人地址，TPL 会自动将众筹通证交付给发送者 - 所有这些都无需信任第三方。

参与者可以使用 TPL 提供的分布式交易所，直接在交易所交易通证以获取其他类型通证或根源卡 BSTK，而无需第三方交易所。

3.16.3.2 协议设计

出埃及地址

TPL 协议在块链中有一个类似的起点，称为“出埃及记地址” - 根源链地址，出埃及地址届时会在根源链官方网站发布，敬请留意。

分发按如下方式进行：

任何在约定日之前将根源卡发送到出埃及地址的人都被协议认可为拥有若干倍数量的哈耶克种子（Hayek Seed, HS）。 例如，如果用户 A 在 12 月 31 日之前向出埃及记地址发送了 100 个根源卡，我的根源卡地址在 12 月 31 日之后拥有 100*N 个 HS。

早期买家获得额外的哈耶克种子。 为了激励，越早购入者激励越多。

再确定关闭购入日之后购入者，系统不再受理，所有打入出埃及地址的资产都会退还。

OP_RETURN

根源链有一些普通用户较少使用的高级功能（如脚本编写），这些功能可以让根源链不断获得更多的新能力，但 TPL 协议通常完全不使用这些高级功能，并且通常也不需要它们在区块链中嵌入数据。

- TPL 协议最初被指定使用根源链普通地址（1 类）在区块块链中嵌入数据
- 后来将数据嵌入根源链多签名交易（2 类）中

- 当前开始支持新的 OP_RETURN 操作码作为根源链核心客户端的一部分（3 类）

交易无效的约定

并非每个根源链钱包都可以让您在付款时选择根源链来自哪个地址，而 TPL 交易必须全部来自持有哈耶克种子的地址。 如果根源链钱包包含存储在多个地址中的根源卡，则用户（或 TPL 协议软件）必须首先确保将发送 TPL 交易的地址在根源链中具有足够的余额来创建交易。 然后，可以从该地址成功发送与 TPL 相关的交易；因此，必须使用支持 TPL 的钱包，或者满足官方 TPL 协议的第三方实现的钱包，才能够安全的发送交易。

TPL 协议交易字段

字段名	大小
通证标识符	
系统	
整数	8Byte
整数	4Byte
整数	1Byte
整数	2Byte
列表标识符	预留
根源链地址	
通证数量	
属性	
响应	预留
字符串	<=255Byte

区块时间间隔	
UTC 日期时间	
时间段	预留
挂单	
MetaDEX 挂单报价	
交易类型	

3.16.3.3 特性

- 区块链浏览器
- 在根源链网络查看哈耶克种子交易
- 查找哈耶克种子资产信息
- 在分布式交易所查看资产交易情况
- DEX 分布式交易所
- 完整功能桌面钱包
- 根源链核心客户端的附生层，不产生任何影响根源链本身的作用
- 支持 MacOS、Windows、Linux 三大平台
- 原生、跨平台用户接口
- 点对点的分布式交易所

3.16.4 根源链虫洞协议及基于根源链的独立资产发行

3.16.4.1 定义

根源链虫洞协议是一种新的智能合约方案，目的是为了让资产的发行更加简化、安全、稳定。它基于根源链实现，在不改变现有根源链共识规则的情况下，使得根源链区块链实现通证的发行、转移和燃烧等基本功能。

基于根源链发行的通证，其交易信息被写在 OP_RETURN 上，其通证的生成、燃烧以及转移都需要根源链原生交易完成。识别 OP_RETURN 里的数据才能够完成对于通证的发行，转移和燃烧。根源链虫洞协议复用了根源链基础设施的交易转账系统。

3.16.4.2 方案

哈耶克种子（Hayek Seed，HS）是协议中的基础货币，其生成是通过燃烧生成(Proof-of-Burn)的机制，用户可以通过向特定地址发送根源卡来获得 HS。这种兑换是单向的，所以通常情况下无法用哈耶克种子兑换 BSTK，哈耶克种子可用于市场自行交易。

新创建的通证需要收取若干 HS 的手续费，手续费会自动燃烧，总供给会减少，给基于 TPL 协议创建的某种类型通证发“空投”需要支付手续费。

增发模式

- 固定通证模式。特点：预挖所有；不能增发，不能燃烧；不能众筹。
- 可众筹通证模式。特点：创建后，自动众筹；创建者不拥有所有通证；众筹结束后，未众筹完的通证自动转入创建者地址；不能增发，不能燃烧。
- 可管理通证模式。特点：创建时，通证数量为 0；不能众筹；可以增发，可以燃烧。

3.16.4.3 优势

根源链通证协议层（TPL）作为一种智能合约方案，最大的优势在于没有改变共识规则，也就是说，该协议对于根源链区块链没有安全方面的影响。

其它的根源链智能合约方案，要么需要修改共识协议，要么需要硬分叉，这对根源链稳定运行本身就是一种挑战，因此，方案就必须越安全越好。

3.16.5 应用

虽然以太坊的一个常见用例是创建不同类型的通证，例如 ERC20 和 ERC721，但根源链 TPL 协议层能够让这一过程更加简化、安全、稳定。

数字货币

数字货币资产是一种以数字形式提供的通证，具有物理对应物相似的属性，但具有底层加密货币系统的好处，例如没有中间方的参与，不可逆交易，或跨境转移等优势。

权益与融资

权益通证可以代表资产的所有权，例如债务或公司股票。

业务通证不是为投资设计，而是提供对产品或服务的预先融资。

数字对象

通证被用来代表其他数字对象，例如社区虚拟游戏卡、文玩字画度量，这些类型的通证是不可伪造的，具有已知的固定供应量以及借助底层区块链的属性，达到可审计的所有权数据存证。

投票

数字通证可用于验证和审计的投票场景。

访问流量定价化

通证控制访问是一种新的概念，基于用户在他或她的钱包中持有的通证来确定对参与到根源链生态中的系统提供访问流量的阈值限定、级别确定等。

3.17 BLFS 架构

3.17.1 BLFS 网络架构 BLNS

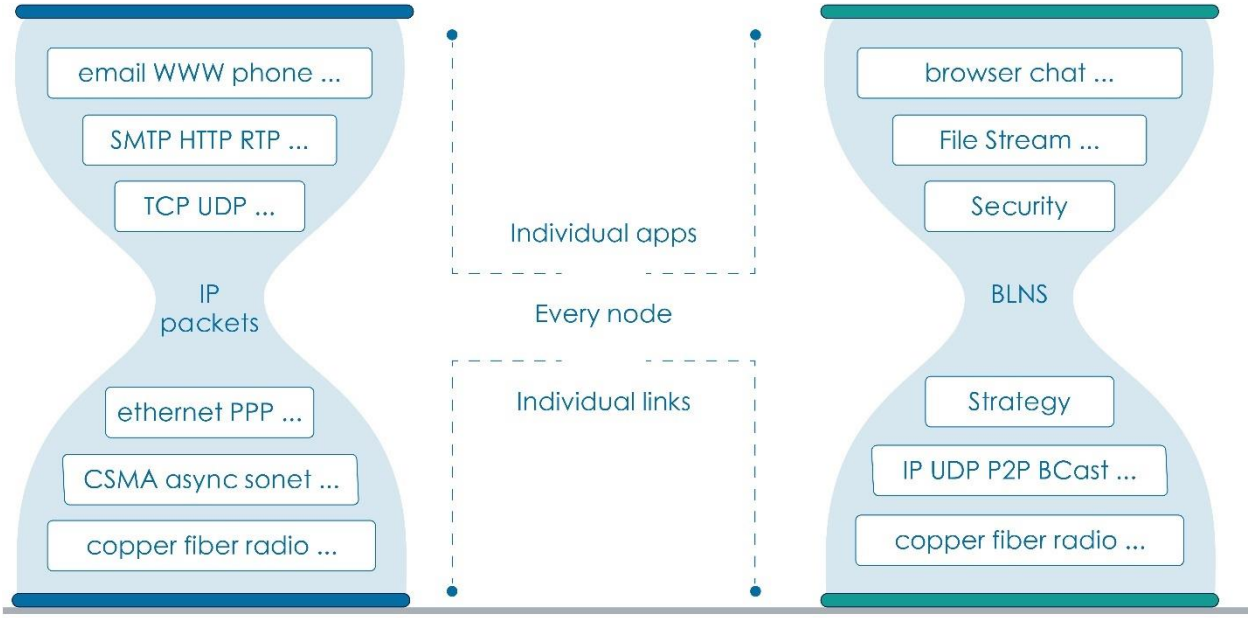
BLP2P 是一个基于分布式存储思想的网络堆栈和库模块化的项目，同时也是根源链的一部分，提供给其他根源链生态组成部分使用。在过去的 15 年中，构建大规模的点对点系统一直是复杂和困难的，BLP2P 是解决这个问题的一种思路与方法。 它是一个协议模块组件，应用程序只需要依赖 BLP2P 就可以完成大量相关功能。 BLP2P 源自 BLFS，是根源链生态中的子项目，能够用于许多不同的项目。

BLP2P 与传输层无关，因此它可以在任何传输协议上运行。 甚至可以不依赖于 IP 层工作，从而形成全新的安全或匿名网络。

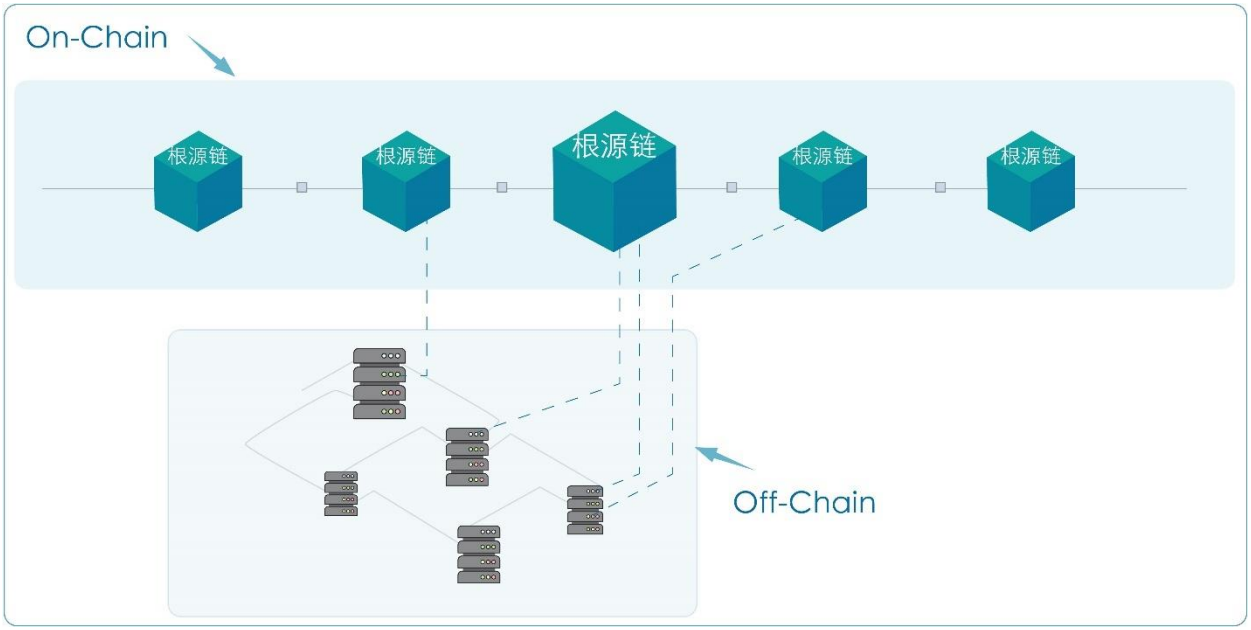
BLNS 采用“蜂腰”架构，从而保证最初的网络设计的健壮与精巧。在这个层次上，它以通用网络层（IP）为中心，实现全球互连所需的最小功能。

在网络传输层面上，BLNS 保留并扩展了路由和转发平面分离的设计原则，从而允许转发平面在路由系统不断发展的同时发挥作用；它允许使用最佳可用转发技术部署 BLNS，同时又并行执行新的路由系统研究。在架构安全性的考虑中，BLNS 通过签署所有命名数据，为蜂腰提供基本的安全构建。

网络流量必须是自我调节的。流量均衡的数据传输对于稳定的网络运行至关重要。由于 IP 执行开环数据传输，因此 BLNS 修改了传输协议以提供单播流量平衡。



3.17.2 BLFS 存储架构



整个 BLFS 架构分为两部分：

On-Chain 部分，分布式存储(On-Chain，链上)有别于分布式存储(Off-Chain，链下)，因为链上数据是稀有的，所以链上数据具备特殊性。根源链 BOS 针对链上数据进行了独特的设计，主要依赖的技术包括 OP_RETURN 和 MultiSig。其存证共识凭证为 PoE（Proof of Exist）。

Off-Chain 部分，根源链采用了新型存储技术。接入层的实现并结合分布式转发的命名规则，实现了基于内容的分布式存储(Off-Chain)服务实例。使用 BLFS 新型存储的所有文件名称都是从其内容的散列中生成的，并意味着同一个文件在每台计算机上都具有相同的名称，并且更改文件内容会导致文件名称的更改。当从服务器下载一个文件夹时,可以根据服务器提供的内容重新计算文件名称来验证文件是否为所请求的文件，并使用 PoE 进行审计和确认。

4 应用场景

4.1 泛物联网 IoT

4.1.1 实物溯源

基于根源链提供的记账和分布式数据存储基础设施，结合物联网线下数据采集和核心功能，为溯源、电商、数据确权等业务场景应用，提供线上线下完整的基础支撑平台，其中重要的业务内容及产品系统如下：

- IoT 设备分类及标示

涉及到的 IoT 设备包括气象站，墒情系统，RFID 芯片，智能控制系统，GPS，图像识别；手持智能终端等；

针对这些设备根据业务及应用的场景，进行分类并进行标示，并建立对应设备采集到的数据序列和集合，相关信息的标示和处理是基于根源链的 BSTK，关键性信息经技术处理后存储到根源链分布式网络。

- IoT 设备的采购及其与业务对象的绑定

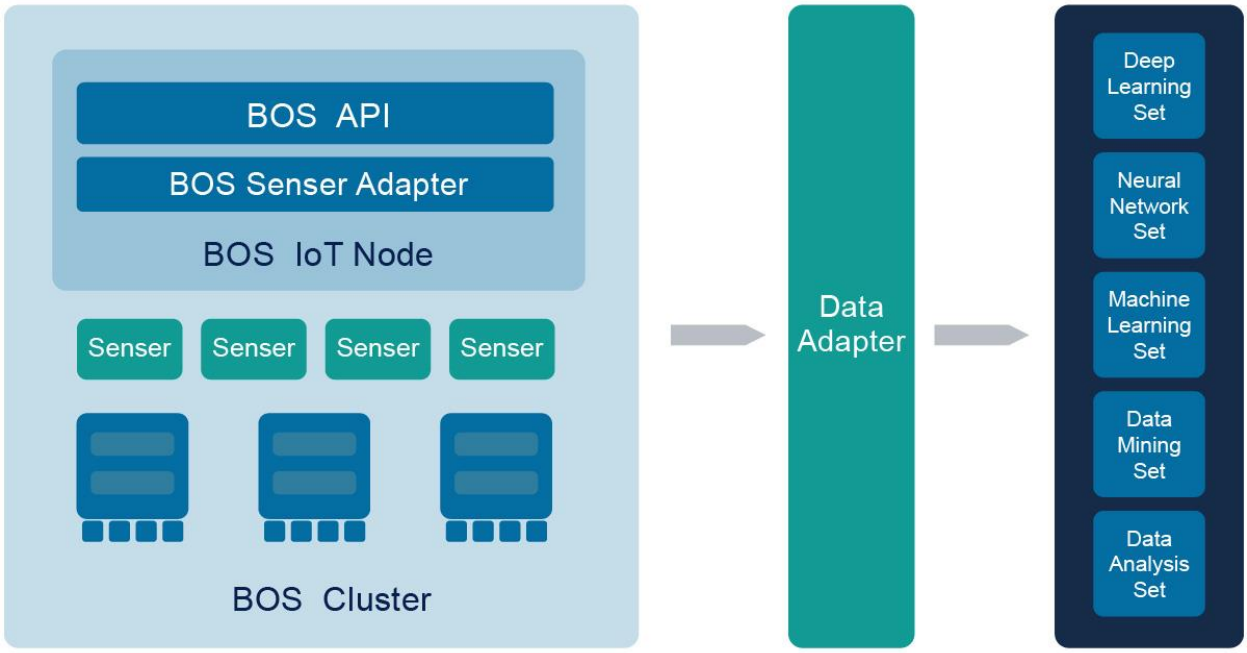
相关的设备结合区块链技术进行相关的分类处理后，供相关的商户采购，这些采购的 IoT 设备预期生产场所、加工包装、仓储物流等相关设施及资产进行管理或绑定，实现对应业务环节业务数据的信息化处理和数据上链。其中，IoT 设备的采购、业务对象绑定、资产及数据确权等在时间和空间维度上数据的处理，使用 BSTK 进行标示或计量，根据业务需要部分数据技术处理后会存储到根源链分布式网络体统。

- IoT 设备数据采集及处理

被 IoT 设备监控的对象（生产资料，产品等）在生产环节、加工包装环节、仓储物流等环节的在时间上的延展，和做空间上的变化，其中和各业务环节相关及用户关心的数据，依次被线下 IoT 设备采集并不断通过各软件节点处理，相关的数据在根源链分布式网络系统不断累积。

4.1.2 车联网

基于根源链的信息确权向汽车提供数字身份，根源链就可以利用 GPS 追踪汽车获取的输出数据，在区块链上给它的位置添加时间戳，进而利用区块链相关的功能属性。消费者、应用开发者和制造商就可以合理利用与分享汽车联网获得的数据。



在根源链的车联网-无人驾驶相关应用中，按照行业和行业边界的构想，可以在链上定义无人驾驶的静态和行为数据结构，并基于数据构建合约来代替合同，从而确保后继的平台训练数据的完备、准确、安全性。此时结合 Token 激励体系，对无人驾驶的各个环节参与者积极性，都可以进行有效的激励。

4.1.3 无人机

无人机是典型面向技术的 IoT，更确切的说是面向人工智能(AI)，包括机器学习(ML)、深度学习(DL)以及神经网络(NN)，它们都有一个核心共通点：数据。结合区块链

的无人机技术，无疑是规范化，可控制化的，并且符合管理需求。例如，无人机与自然人身份的溯源，可以避免非现场犯罪的可能。

对此类或者类似的应用，根源链提供一套完整的 IoT 智能合约，智能合约模块属于 BOS D-Right 模块，提供桩与代理接入模型，配对 BOS 节点通过代理获取合同的具体细则，以及合同的控制 API。

4.2 公证

传统的公证行业在数据的存储和使用方面主要以中心化方式实现，不仅维护成本高，而且不利于公证业务的市场拓展，主要表现在用户端的存证、取证、公证等业务环节。

根源链解决了公证行业的存证取证在传统大数据平台、分布式平台中的数据不可靠、易篡改、不可信、权益第三方等问题；通过 BOS 节点设备提供的一整套方案组合，不仅满足了数据存储的需要，而且结合了最新最全的区块链技术作为支撑

根源链解决了公证行业的取证所签署的电子存证指纹的中心化、存证孤岛问题；通过使用区块链存证 ID 作为指纹数据，不仅确保存证数据的去中心，而且确权数据随着时间的推移越来越牢固。

4.3 分布式电商

基于区块链技术的分布式电商作为一种全新的在线信息分发共享和交易系统，既为区块链分布式网络上的各个节点提供点对点的信息发布、产品检索、交易及确权等平台支持，也为基于传统在线交易系统提供溯源产品、商品防伪、信息保全、交易记账、数据确权等服务。分布式电商核心组成部分如下：

- **在线交易系统**

在线交易系统作为整个分布电商软件的载体，会发布 PC、IOS、Android、H5 及小程序等版本。

- **商品检索系统**

分布式电商系统商品的信息来自溯源系统，并通过溯源系统接口获得商品的厂家、生产

制作包装、物流等溯源 信息；

- **交易及记账系统**

分布式电商的交易及记账基于区块链产品系统，目前版本使用根源链系统，今后计划推出基于比特币及以太坊的版本；

- **用户账户体系**

在分布式电商体系内商品交易计价单位的源点及用户资产数量计算，是基于根源链的 BSTK，源点按比例与 BSTK 映射；

根源链计划发布一个轻量化 DEMO 应用：买卖大集。买卖大集作为分布式电商的一种形式，其在前端商品信息推送、营销活动等功能方面采用传统电商的处理方式，前端是一个 H5 网站，后端商品、交易及结算、账户体系则是基于根源链系统。

4.4 信息数据溯源确权服务

根源链溯源系统使用 IoT 物联网设备进行数据采集和监控，并通过根源链系统实现数据存储，实现产品信息的可追溯，不可篡改，为各种业务应用场景提供基于分布式区块链网络的数据存储和时空数据溯源的功能。

4.4.1 溯源系统的基础

- **IoT 物联网硬件设备**

数据采集和跟踪监控设备：气象站，土壤墒情系统，RFID 芯片，GPS 设备，图像及识别设备，声纹及识别设备，加速度等传感器

前端数据处理及智能前端机：区块链计算机，智能控制系统，手持智能终端

- **区块链分布式记账系统**，早期版本使用根源链实例

4.4.2 溯源系统对外提供的服务

- **商品溯源**

对商品的生产环节、包装鉴定环节、物流环节、交易和消费环节进行相关的信息的追溯。

- **数据确权**

基于 IoT 物联网设备及区块链分布式网络系统，追溯和记录各个业务环节的关键信息，并对相关的实物商品及相关的数据进行权责确认，并形成确定数字标签进行对所有权进行数字化。

4.4.3 溯源系统对外服务的提供方式

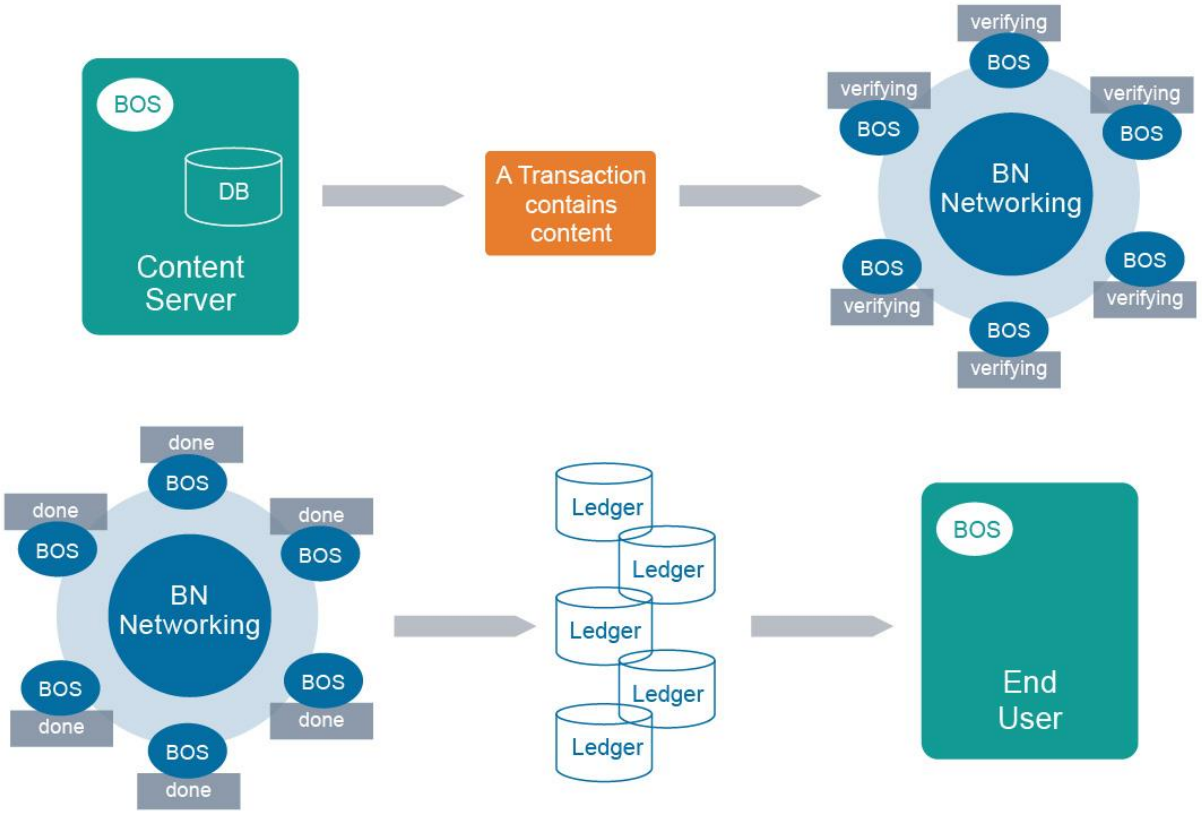
- API 接口
- SDK（Android 及 IOS 版本）
- H5 网页

4.5 多媒体内容服务

多媒体分发也称为内容交付。根源链的多媒体内容服务基于各类基础资源与 BLFS 等存储相关的技术特性，且支撑所有数据的加密安全、共识、转发以及存储。

作为一种多媒体内容的数字分发形式，其所描述的多媒体主要包括音频，图像和视频。与传统处理方式相比，例如专用富媒体应用服务器、基于 CDN(内容分发网)的云端 DaaS，甚至分布式流媒体服务，这些方案没有一个解决侧重于交付内容的安全性和完整性等问题。

业务层面的区块链媒体服务过程：



- 服务端接收到资源请求以后发送出资源，并且包含一笔 Transaction；
- 接下来通过业务层 网络每个 BOS 端的共识验证；
- 通过所有 BOS 端共识之后，将这笔 Transaction 记账并汇总于总账本；
- 最终，用户端接收到安全和完整的媒体数据。

4.5.1 实时版权保护算法

根源链可以实现对多媒体的版权保护，并计划基于 API，针对此种需求进一步提供一种分布式实时水印系统，当服务节点接收到来自用户的请求时，通过实时水印处理之后向用户发送期望数据。

4.5.2 根源链多媒体平台

- 根源链引入了相位算法，并结合块嵌套和自嵌入水印的概念；
- 根源链提供了可信赖的机制，以及分布式内容框架；
- 智能合同促使了媒体信息承接双方的行为约定；
- 在根源链强有力的支撑下，基于小波的自嵌入水印算法解决了内容不完整、安全检测不均衡、篡改、失效销毁等问题，实现了一种独具创新的多媒体平台。

4.6 其他应用场景

4.6.1 DApp

分布式应用程序或“DApps”是大家较为熟悉的区块链应用概念，而根源链基于结合物联网的公链基础设施特性，可以支持，尤其是支持和物联网相关紧密的各类 DApp。通过根源链提供的各类基础与合约 API，DApp 可以更容易的将后端部署在区块链上，并依靠根源链提供的计算、网络、存储、加密能力来向后端提供完整的运行环境。

目前，DApp 主要以积分类型的轻量化应用为主。

4.6.2 DCC 分布式云计算

与 DApp 类似，由于根源链可以提供计算、网络、加密、存储等基础资源并通过 API 等多种方式来调用，并且能够形成规格化调用，同时又能通过 Token 机制来协调更多的 Off-Chain 同步/异步资源，因此适合作为分布式云计算基础设施使用，例如解决并行计算。

此外，根源链具备资源的分布式调度特性，大量的碎片化资源对于去中心化的网络来说最为理想，不仅可以帮助解决基础资源服务提供商的服务滞后问题，而且可以有选择性地实施数据的同步和过滤。

4.6.3 政务

作为公链，根源链的透明度和不可篡改有助于保持政务信息和执行监督的透明与公正，以及政府形象的维护。

基于区块链技术提供政务应用或政务公开记录并不是根源链的首创，但根源链在政务方面的应用，相比其他私有链、联盟链，所提供的能力与应用更为多样、全面。

具体表现为：

- 合约机制可以用于一些需要公开和强制执行的事务；
- 信息上链记录可以用于公告；
- BLFS 存储在法院存证、交通证据存证方面提供远高于其他区块链解决方案的存取能力和读写速度，同时提供足够高强度的防篡改；
- 基于根源链的计算能力和可调度加密能力，可以作为政务网的安全应用网关等等。

4.6.4 大数据与人工智能

海量获取任何类型的大数据来源对任何想要进行机器学习的人来说都是一项挑战。基于根源链的 Token 激励机制，可以对提供数据与计算等基础资源能力的对象进行激励，而让需要数据进行机器学习的使用者参与进来并使用 Token 交换。在参与者的生态圈中，提供者的数据转化为 token 再转化为等价交换物，使用者输入等价交换物获得 token 再转化为需要的数据。相比传统的数据采集模型，根源链无疑提出了一种新的范式，不仅有利于该范式下的参与者，而且促进了平台数据获取的永续性，进一步加强了根源链生态建设的活力。

根源链改变人工智能的途径：

- 人工智能技术的发展依赖于大量来源的数据可用性，根源链作为公链，天然具备公开、透明的数据提供者角色
- 人工智能的发展取决于数据的获取方式永续性，根源链作为永续的数据源，是机器学习进行数据训练的福音，彻底解决了数据匮乏的问题
- 人工智能的发展取决于数据的流动性，根源链作为公链的分散性意味着人工智能的发展不再束手束脚
- 人工智能的发展取决于数据的可信性，根源链比封闭的人工智能有更高的透明度利于审计，而且新型的密码学实现促使数据更加安全可靠
- 人工智能的发展取决于数据的安全性，根源链的算力共识机制保证了基础设施固若金汤，确保数据安全、可靠

5 经济原型

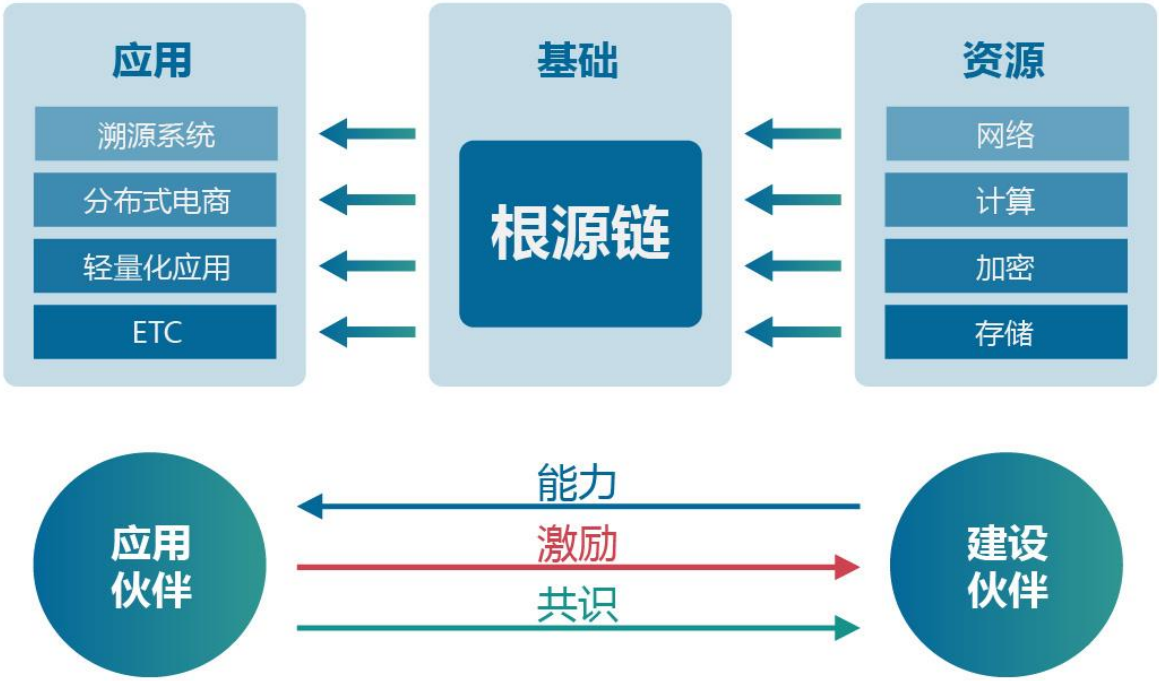
5.1 经济体系

根源链网络基于参与根源链生态体系建设的各方所提供的资源与能力，通过抽象与运算，记账生成根源链经济体系通证 BSTK（BlockChain Source Token），其正式名称为根源链通卡，简称为根源卡，BSTK。

BSTK 总量上限为 210 亿个，约 147 年生成完毕。实际流通的 BSTK 数量受根源卡驱动行为机制限制，总是小于甚至远小于当前已生成的 BSTK；基于同样原因，BSTK 的实际总量亦小于总量上限。

5.1.1 通用经济体系

根源卡 BSTK 不是单纯的代币或者数字加密货币。根源卡是根源链网络中用于各类资源贡献与奖励的结算工具，亦是根源链的数据溯源与确权的基本载体，同时具备经济价值与使用价值。在经济体系中，根源卡 BSTK 是根源链生态中不可或缺的组成部分。



（1）生态体系

根源卡是根源链所使用的唯一通行的加密数字令牌，它在各类参与根源链生态的不同角色用户之间建立经济关联，通过基于根源卡为结算工具的交易行为，使能力与激励的计算形成公平共识，简化交易环节，促进经济协作发展；通过使用基于根源卡驱动的各类链上应用，实现 Off-Chain to On Chain 关联，实现区块链化的应用生态，共同建设可信的技术与经济体系。

（2）建设伙伴

为了构建完整的生态体系，根源链需要对参与根源链生态建设，并提供基础资源的建设伙伴提供公平奖励。建设伙伴付出和使用自己的资产、设备、资金、资源，通过各类通用或定制型的软硬件及物联网设施，向根源链提供网络、计算、加密、存储等基础性资

源，甚至基于区块链计算机和算力服务进行优化，提供一次加工型资源，而根源链则基于相应的共识机制向建设伙伴给予根源卡奖励。建设伙伴及最终用户均可以将获得的根源卡用于各类使用根源卡驱动的应用中，或用于兑换根源链上各类应用伙伴提供的增值服务。

（3）应用伙伴

通过建设伙伴提供的各类实体或数字化的资源，及其通过根源链 BOS 转化后形成的支持能力，根源链逐步形成强大的生态体系；并基于这部分基础资源，为应用伙伴提供可靠的公有链基础设施能力支持，为整个商业经济带来活力。 应用伙伴基于根源链这一基础设施，搭建各种商业应用，实现链上与实体经济的双重获益。应用伙伴使用根源卡获得相应的资源，以及使用基于根源卡驱动的业务流，低成本的建设自己的商业应用，同时将具体的服务提供给最终用户，获得根源卡或法币资金收益。

5.1.2 链上经济体系和根源卡驱动行为机制

在根源链生态体系中，根源卡 BSTK 不仅是通用经济体系中的结算工具，亦在和区块链紧密结合的环节中，承担技术-经济的双重交互身份。

（1）记账挖矿行为

使用定制的区块链计算机、算力服务等定制型的资源提供设施，用户可以快速成为根源链建设伙伴，向根源链网络提供网络、计算、加密、存储等基础资源，这一过程接近于传统的数字货币挖矿；用户将通过获得新生成的根源卡 BSTK 的形式，获得基础的记账挖矿奖励。

（2）根源链使用行为

应用调用根源链 API，进行数据查询、统计、分析、计算，以及在特定合约机制下运行 DAPP，交易转移 BSTK 等，这些过程相当于将 BSTK 作为 GAS 使用，会消耗一定数量的根源卡，并由对应的资源提供方在记账挖矿过程中，基于共识计算获得。

（3）根源卡驱动行为

大多数基于根源链的应用，其基本的数据信息传递载体需要使用相应数量的根源卡 BSTK。这一过程中，所使用的根源卡将以类似锁定的状态限制在应用交互环境内，并在遵循上文的消耗方法的前提下，根据应用自身特性，存在根源卡销毁机制。

例如，在某一食品溯源应用中， 基于应用伙伴企业 A 的商业考虑，使用一定数量（e.x：1000 万个 BSTK）的根源卡进行产品溯源。则在企业 A 的食品溯源应用持续运行过程中，该 1000 万 BSTK 会被 Label 染色等机制锁定在本应用内，无法流通至二级市场，也不能和未作 Label 处理的 BSTK 混用；同时，根据企业 A 的产品特性，这一部分 BSTK 在应用运行结束后，会有一定比例被永久销毁。

不仅在商业应用中，这一行为机制同样广泛适用于基于根源链生态的各类应用环境与应用级别。例如，在基于根源链的 LN 交易环境中，若节点服务作为固定服务设施长期存在，则其根据交易上限锁定的对应数量的 BSTK（e.x：500 万个 BSTK）同样无法进行流通和交易，但不存在销毁机制。由于这一机制的存在，使用根源卡驱动行为的应用越多，根源卡的实际可流通数量越少，总量上限也会因此减少。

5.2 通证分配

5.2.1 通证比例

根源卡 BSTK 总量为 210 亿。基于根源卡 BSTK 并非一次产生，大部分是由记账挖矿生产而来，因此以下通证分配比例均 BSTK 总量的比例计算，实际流通数量远小于该总量。部分分配比例存在锁定，见通证方案一节所述。

由于根源卡驱动行为机制的存在，根源卡 BSTK 不存在固定的流通比例。任意时刻实际流通量可按如下方式进行计算：

当前流通量=当前记账挖矿生产总量-当前动态锁定量+解锁流通量

根源卡 BSTK 的总分配比例为：

记账挖矿生产：70%

-动态锁定：52%

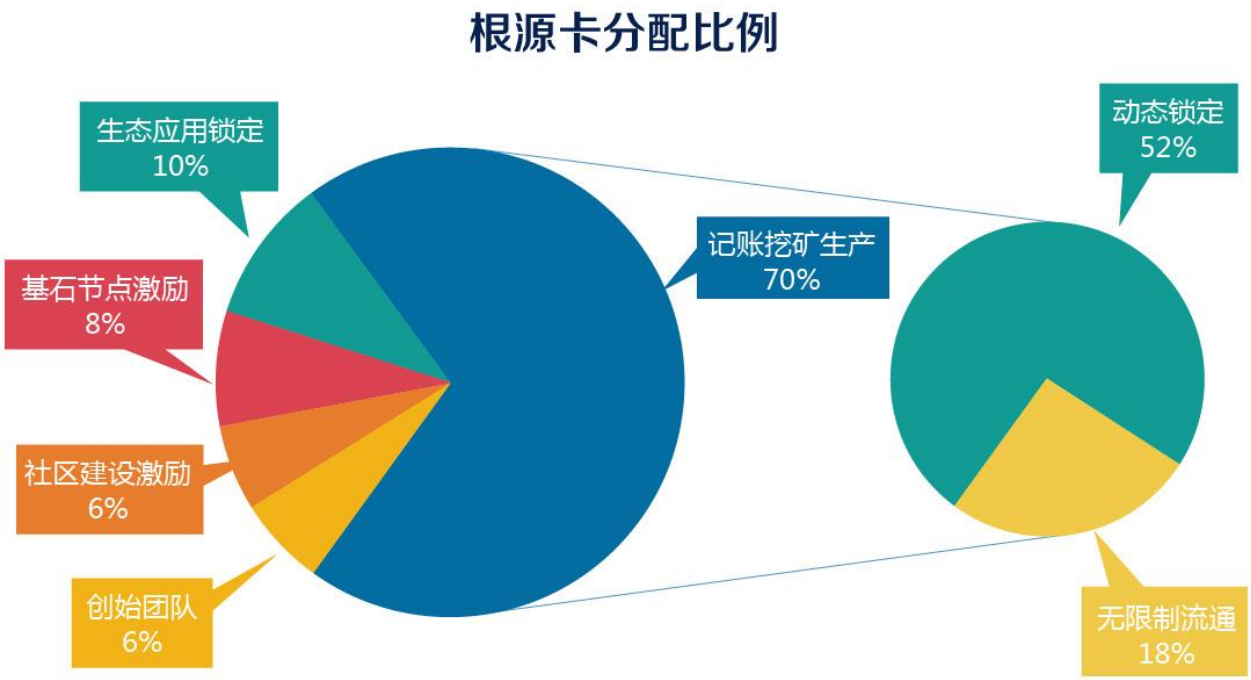
-无限制流通：18%

生态应用锁定：10%

基石节点激励：8%

社区建设激励：6%

创始团队：6%



5.2.2 通证方案

（1）记账挖矿生产

通过向根源链提供相应资源，从而记账挖矿生产是根源卡 BSTK 的直接获取方法。挖矿记账生产的根源卡占总量的 70%，即 147 亿个，平均每年产生约 1 亿个，147 年左右产生完毕。实际产生速度受根源链网络规模与应用规模影响，约在 140-210 年之间。任何向根源链生态提供相应资源的用户，均可以成为建设伙伴，通过此种方式获得 BSTK。

在这一方式获取的 BSTK 中，会锁定最多占当前已通过记账挖矿生产数量的 75%（最多占根源卡总量的 52%）的根源卡，通过根源卡驱动行为机制，动态锁定于根源链应用生态中，无法进行不受限制的流通和交易。为了保证根源卡的通用经济体系正常运作，根源链将保证最低有占当前已通过记账挖矿生产数量的 25%（根源卡全部生产完毕时，最少占总量的 18%）的根源卡属于无限制流通的根源卡。由于记账挖矿生产的特性和以上条件，根源卡初始无限制流通的数量为 0，随着记账挖矿行为，逐年提升。

（2）生态应用锁定

根源链是为应用服务的基础设施公链，为促进应用生态发展，永久锁定总量的 10%，即 21 亿个，用于根源卡驱动行为的应用使用。和动态锁定的部分不同，该部分锁定的根源卡永不进入无限制流通，仅在各生态应用内部的相关交互环境中使用，作为信息数据传递和记录的凭证。

由于生态应用对根源卡的实际需求大于 21 亿个，因此实际环境中，挖矿记账中的动态锁定部分会动态补充应用的需求。实际应用中，被锁定而无法流通的根源卡所占的实际比例最大是 62%，即 130.2 亿个。该数量会随着挖矿记账产生根源卡而动态变化。

（3）基石节点激励

为了根源链基础资源能力快速达到相应技术所要求的基准，根源链提供占根源卡总量的 8%，即 16.8 亿个，用于基石节点激励。在根源链的不同发展阶段，使用各类基于根源链网络技术优化要求的网络、计算、加密、存储、物联网等综合资源设施，持续参与和推进根源链网络快速发展的用户，会根据其资源类型和节点贡献，获得额外的基石节点激励。该部分存在锁定，自奖励确认后第 30 日起，分 N 日（最长 360 日）发放，每日发放 1/N，直至发放完毕。获得奖励的用户中途不符合奖励标准的，后续奖励停止发放。

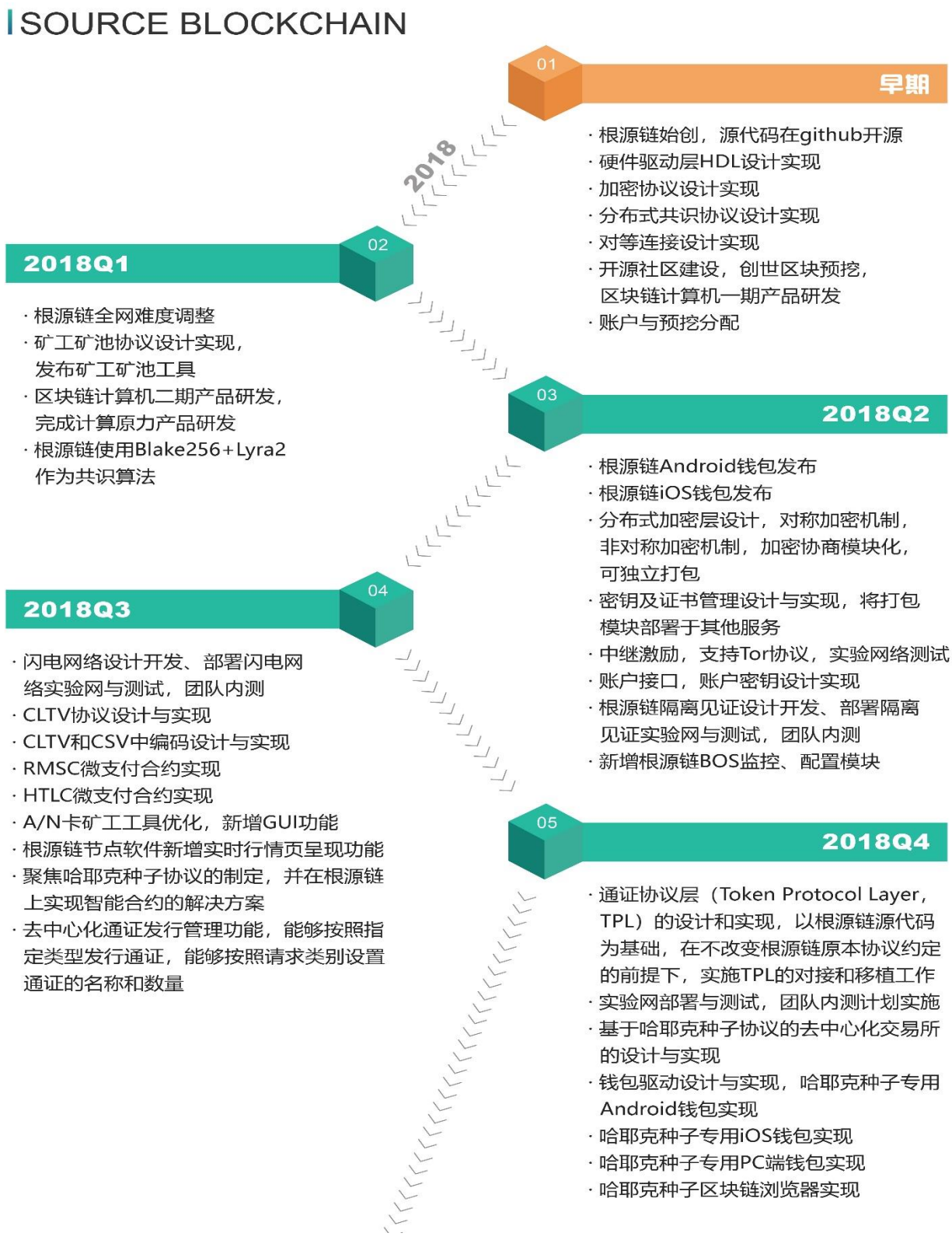
（4）社区建设激励

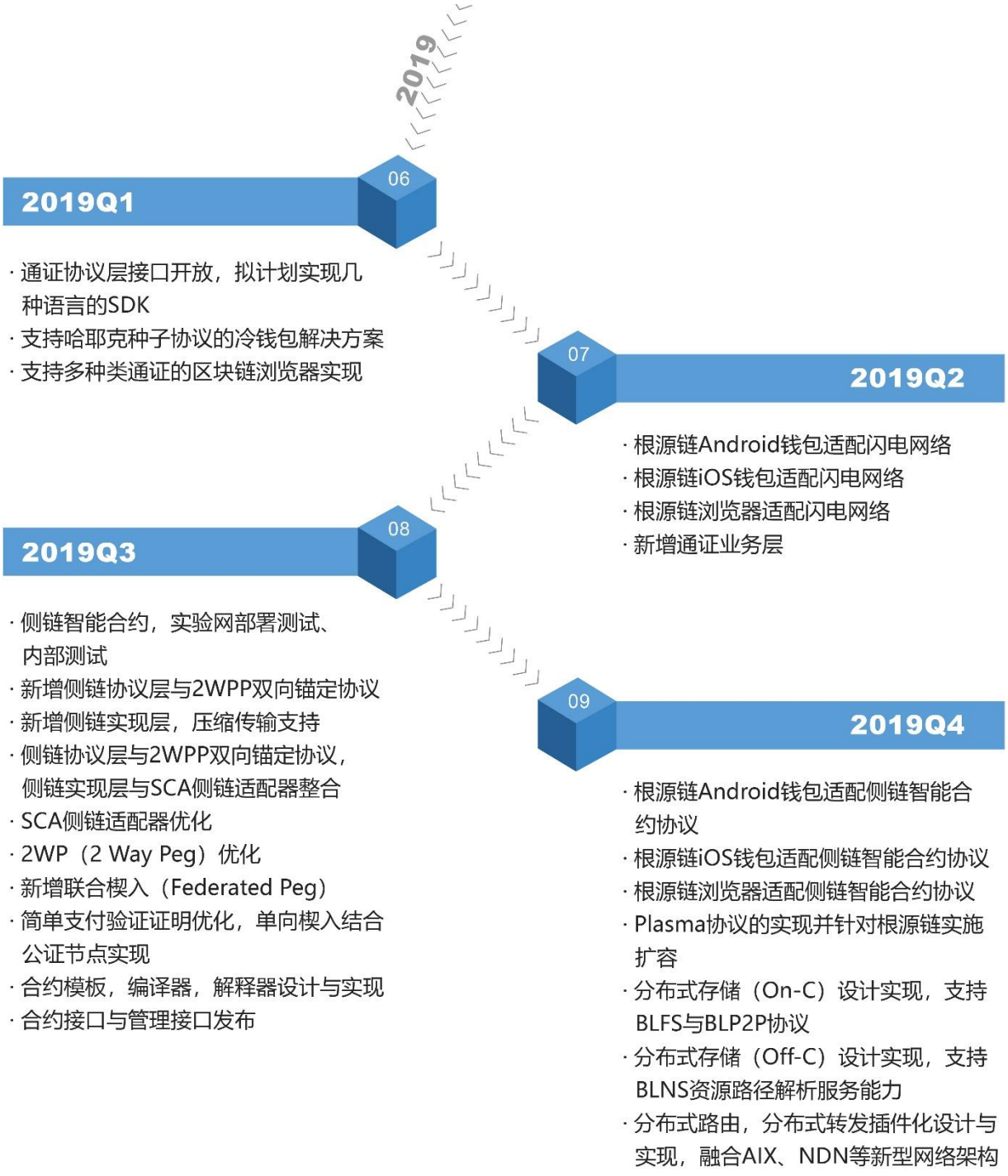
根源链需要社区的力量以快速发展和完善。根源链提供占根源卡总量的 6%，即 12.6 亿个，用于社区建设激励，包括但不限于支持和孵化各种基于根源链的 DAPP 及应用生态，额外奖励早期记账节点和重大贡献，作为社区合作，项目合作经费，推进生态样板应用发展，增强根源卡流通性与价值管理等使用。该部分存在锁定。对于个人或团队所获得的奖励性质的根源卡，自奖励确认后第 6 个月起，每 6 个月释放 20%，直至释放完毕。

（5）创始团队

根源链将提供占根源卡总量的 6%，即 12.6 亿个，用于奖励创始团队前期对于根源链项目的探索和开发，激励其长期维护与发展根源链。该部分存在锁定。自主网上线后第 6 个月，第一年解禁该部分 40%，第二年解禁 20%，第三年解禁 20%，第四年解禁剩余部分。

6 技术路线图


SOURCE BLOCKCHAIN
01
早期
· 根源链始创，源代码在github开源
· 硬件驱动层HDL设计实现
· 加密协议设计实现
· 分布式共识协议设计实现
· 对等连接设计实现
· 开源社区建设，创世区块预挖，区块链计算机一期产品研发
· 账户与预挖分配
2018
02
2018Q1
· 根源链全网难度调整
· 矿工矿池协议设计实现，发布矿工矿池工具
· 区块链计算机二期产品研发，完成计算原力产品研发
· 根源链使用Blake256+Lyra2作为共识算法
03
2018Q2
· 根源链Android钱包发布
· 根源链iOS钱包发布
· 分布式加密层设计，对称加密机制，非对称加密机制，加密协商模块化，可独立打包
· 密钥及证书管理设计与实现，将打包模块部署于其他服务
· 中继激励，支持Tor协议，实验网络测试
· 账户接口，账户密钥设计实现
· 根源链隔离见证设计开发、部署隔离见证实验网与测试，团队内测
· 新增根源链BOS监控、配置模块
04
2018Q3
· 闪电网络设计开发、部署闪电网络实验网与测试，团队内测
· CLTV协议设计与实现
· CLTV和CSV中编码设计与实现
· RMSC微支付合约实现
· HTLC微支付合约实现
· A/N卡矿工工具优化，新增GUI功能
· 根源链节点软件新增实时行情页呈现功能
· 聚焦哈耶克种子协议的制定，并在根源链上实现智能合约的解决方案
· 去中心化通证发行管理功能，能够按照指定类型发行通证，能够按照请求类别设置通证的名称和数量
05
2018Q4
· 通证协议层（Token Protocol Layer，TPL）的设计和实现，以根源链源代码为基础，在不改变根源链原本协议约定的前提下，实施TPL的对接和移植工作
· 实验网部署与测试，团队内测计划实施
· 基于哈耶克种子协议的去中心化交易所的设计与实现
· 钱包驱动设计与实现，哈耶克种子专用Android钱包实现
· 哈耶克种子专用iOS钱包实现
· 哈耶克种子专用PC端钱包实现
· 哈耶克种子区块链浏览器实现


2019
06
2019Q1
· 通证协议层接口开放，拟计划实现几种语言的SDK
· 支持哈耶克种子协议的冷钱包解决方案
· 支持多种类通证的区块链浏览器实现
07
2019Q2
· 根源链Android钱包适配闪电网络
· 根源链iOS钱包适配闪电网络
· 根源链浏览器适配闪电网络
· 新增通证业务层
08
2019Q3
· 侧链智能合约，实验网部署测试、内部测试
· 新增侧链协议层与2WPP双向锚定协议
· 新增侧链实现层，压缩传输支持
· 侧链协议层与2WPP双向锚定协议，侧链实现层与SCA侧链适配器整合
· SCA侧链适配器优化
· 2WP（2 Way Peg）优化
· 新增联合楔入（Federated Peg）
· 简单支付验证证明优化，单向楔入结合公证节点实现
· 合约模板，编译器，解释器设计与实现
· 合约接口与管理接口发布
09
2019Q4
· 根源链Android钱包适配侧链智能合约协议
· 根源链iOS钱包适配侧链智能合约协议
· 根源链浏览器适配侧链智能合约协议
· Plasma协议的实现并针对根源链实施扩容
· 分布式存储（On-C）设计实现，支持BLFS与BLP2P协议
· 分布式存储（Off-C）设计实现，支持BLNS资源路径解析服务能力
· 分布式路由，分布式转发插件化设计与实现，融合AIX、NDN等新型网络架构

7 结语

区块链技术将开启全新的经济协作生态。本白皮书主要对根源链当前阶段的整体技术框架以及所涉及的相关基础技术进行了简明的解释，然而随着技术的进步与应用领域的拓展，根源链也将持续不断的更新与进化，相应的，根源链的白皮书也会紧随项目的发展而不定时的进行持续更新。根源链作为一个基于数据信息追溯及确权的泛物联网应用公有链，将不断追求技术上的进步与突破，成为高可用型公链基础设施。根源链团队也欢迎新老伙伴一如既往的支持根源链与根源链社群，共同参与根源链项目建设，参与区块链行业，分享区块链带来的价值。