Data Copyright Blockchain 项目

白

皮

书

编制单位：杭州飞链云科技有限公司

编制时间：二零二二年九月

版本：1.1

目录

第一章 项目背景

从比特币诞生，到智能合约横空出世，再到侧链跨链，区块链引领着几乎每一次重大的技术革新。当前区块链技术被誉为是第四次工业革命代表性成果之一，是最有潜力触发第五轮颠覆性革命浪潮的核心技术。而由于其巨大的应用潜能，包括在数字金融、物联网、智能制造、供应链管理、数字资产交易等多个领域的广泛前景，区块链也越来越受到国际社会的重视，成为各个主要大国间新的竞技场。

正是在于区块链技术前期不断的探索，如今的“区块链+”产业生态的应用场景正在呈多元化趋势发展，从数字金融场景延伸到各实体领域，引领传统行业借助区块链技术在金融、医疗、司法、能源、教育和物流等领域方向创新探索，不断尝试提升服务或升级商业模式。

不难看出，“区块链+”产业生态雏形已现，区块链产业板块内各细分领域发展正在趋向成熟，在不断的竞争与合作中，共同推动着整个产业生态的良性发展。“区块链+”产业创新趋势发展势头正好，全球区块链产业生态都在进一步丰富，百花齐放。

第二章 区块链介绍

2.1 什么是数字资产

数字资产（Digital assets）是指企业拥有或控制的，以电子数据形式存在的，在日常活动中持有以备出售或处于生产过程中的非货币性资产。

广义数字资产是指个人、企业拥有或控制的以电子数据形式存在的资产形式，在日常活动中持有用来交换或行权对应的实物资产。狭义数字资产专指登记在区块链分布式账本上的计算机程序（代币），可以对它进行编程，资产之间的交换是代码与代码的交换。

2.2 什么是区块链

区块链是分布式数据存储、点对点传输、共识机制、加密算法等计算机技术的新型应用模式。所谓共识机制是区块链系统中实现不同节点之间建立信任、获取权益的数学算法。它本质上是一个去中心化的数据库，同时作为代币的底层技术。区块链是一串使用密码学方法相关联产生的数据块，每一个数据块中包含了一次比特币网络交易的信息，用于验证其信息的有效性（防伪）和生成下一个区块。

狭义来讲，区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构， 并以密码学方

式保证的不可篡改和不可伪造的分布式账本。

广义来讲，区块链技术是利用块链式数据结构来验证与存储数据、利用分布式节点共识算法来生成和更新数据、利用密码学的方式保证数据传输和访问的安全、利用由自动化脚本代码组成的智能合约来编程和操作数据的一种全新的分布式基础架构与计算方式。

从科技层面来看，区块链涉及数学、密码学、互联网和计算机编程等很多科学技术问题；从应用视角来看，简单来说，区块链是一个分布式的共享账本和数据库，具有去中心化、不可篡改、全程留痕、可以追溯、集体维护、公开透明等特点。这些特点保证了区块链的“诚实”与“透明”，为区块链创造信任奠定基础。而区块链丰富的应用场景，基本上都基于区块链能够解决信息不对称问题，实现多个主体之间的协作信任与一致行动 。

2.3 区块链和数字资产的融合

在资源数字化后出现了诸多问题，比如盗版侵权、隐私泄露、数据非法倒卖等问题。这些问题背后的关键原因在于，数据资源的交易流转、权属证明、权益保护等机制还并不够完善，使“数字资源”难以形成“数字资产”，数据的价值难以充分显现。

区块链技术的出现解决了上述问题。越来越多的行业正在提出自己的区块链解决方案。它可以落地应用后能迅速发

挥的作用。区块链可以帮助数字资产进一步发展，实现升级。具体如下：

从中心化到去中心化，构建数字资产的生态圈。区块链促进各行各业的产品和文化交流，不再依赖第三方机构或是中心化管理。

从不信任到信任，区块链帮助数字资产解决了欺诈、重复支付等问题。系统运作公开透明，通过“签名”机制和利用“少数服从多数”的原则，可以从机制上保障信用。用户随时可以查看追溯代币的来源，不再担心造假等风险。

从不安全到安全，在每次发生交易后，从当前节点会向所有节点发送信息。当再次交易时，区块会通过其他节点的数据自检数据是否被篡改，一但发现会从其他节点的数据恢复过来，有效的杜绝了黑客对数据的篡改。公开的区块链浏览器，可以使用户下载区块链的相关交易或者区块的签名数据，公开透明的用户监督机制，也保证了内部人员无法对数据进行篡改。

第三章 Data Copyright Blockchain 介绍

3.1 什么是 DATACBc

DATACBc（全称：Data Copyright Blockchain，又称 DCB）是我们对比特币加密算法、智能合约以及共识方面进行优化后的自研区块链品牌，具备高性能、高安全、高可靠、数据完全公开透明的特点。DATACBc BaaS（Data Copyright Blockchain Backend as a Service）为企业提供便捷、实惠、稳定的区块链一站式平台服务。帮助企业在联盟中快速完全区块链网络的部署、监控和运维，有效解决金融、商业、司法等各种行业场景区块链应用问题。

核心算法开源地址：

https://gitee.com/datacbc/datacbc/ （国内）

https://github.com/chenhaoxiang/datacbc （国外）

3.2 产品特点

3.2.1 跨场景/跨云厂商平台

可选择不同云厂商的不同地域节点，依托于可靠云厂商的硬件能力，使用完善的区块链底层协议搭建，支持快速部署和增减节点。自研跨云厂商技术，基于 P2P 技术、BaaS 底层突破服务器地域的限制。实现不同云厂商，不同地域的互通互联。极大提升多企业之间联盟链的协作和灵活性。

3.2.2 轻松接入

提供产品全部能力接口的 API 联盟链公开信息的可视化操作页面快速、安全、灵活接入区块链。大幅降低企业业务开发、迁移及维护工作量，节约成本。一行 shell 代码便可以在一台 Linux 服务器下载运行区块链节点，新节点加入后，会自动通过配置的旧节点同步区块数据。

3.3.3 公开可信/合规安全

白皮书的公开、核心源码的开源、联盟区块链的区块与交易信息的公开，保证联盟链的可信度（任何用户都可以通过节点的区块链浏览器下载或者查看区块上的任何数据）。强大的机构，企业作为 DATACBc 的联盟节点，极大的增加了联盟链以及底层的背书信任；节点之间采用 CA 证书验证级别通讯、使用金融级别加密算法，具备超级计算机无法破解的加密能力全方位保障数据传输与存储的安全可信。通过完善的用户、密钥、权限管理和隔离处理，多层加密保障，可靠的网络安全基础能力，在保障数据信息安全的基础上，同时实现透明化合规监管。

3.3.4 全行业覆盖

提供数字版权、司法存证、防伪溯源、供应链金融等多行业应用场景服务，全方位解决用户痛点，提供整套完善的行业应用解决方案。

第四章 DATACBc 区块链技术应用

4.1 分布式结构

DATACBc 基于 RocksDB，采取分布式网状节点结构（所有基于 DATACBc 运行的节点，共同维护着 DCB 网络），DCB 网络中各结点之间有多条通路，每个完整的节点均在本地存储全部的区块链数据，任何单节点可以接受所有数据的丢失。

分布式网状节点结构没有固定的连接形式。从发信点到收信点的通路不止一条，通信时，由网络根据各结点的动态情况选择通信的实际路径。通信的控制功能分散在各节点上。它是最复杂的一种结构。它的通信控制也最复杂，对分散在各结点上的数据资源的管理也很复杂。由于节点与区块链网络之间存在多条通路，当某些节点与区块链网络链路发生故障时，依然能够保证通信，所以有非常高的可靠性。

分布式记帐：采用分布式记账可以确保帐本信息的安全性和真实性。在区块链网络中，记录历史交易的信息被传递给了每一个节点，每个节点都可以拥有和存储一本完整、一致的交易总帐记录。即使个别的节点帐本被攻击，数据被篡改，也不会影响到全网的总账的安全性。

分布式传播：全网的节点是通过底层网络协议点对点的方式连接起来的，没有单一的中心化服务器。消息通过 P2P 网络层协议，由单个节点直接发送给全网的其他所有节点。

分布式存储：通过分布式传播后，所有数据均存储于各节点的电脑中，并且可以实时更新。相当于把帐本等数据实时共享于所有网络节点。实现了去中心化，有效的避免了单一节点被攻击造成的数据篡改。极大的提高了数据库的安全性。

通过分布式的结构，实现了去中心化，使用 P2P 网络模型。不再需要中央服务器，每一台联网的计算机都是一个独立的个体，通过 P2P 点对点协议连接到其他成千上万的计算机，最终所有接入 DCB 网络的计算机连接成为一个密密麻麻的网，从某一节点上发出的信息，最终可以扩散到全球所有的节点。此结构的优势是即使其中一部分节点发生故障，也不会影响整个网络的通信。

4.2 区块链数据结构

区块链是由包含交易信息的区块从后向前有序地链接起来的一个一个数据结构。它被 RocksDB 存储在文件系统中，使用区块时，再通过反序列化传输字节。每个区块都指向前一个区块。数据结构主要分为三部分：区块头、交易列表和父区块。

对于每个区的块头进行 SHA256 加密哈希，生成一个哈希值，通过此哈希值识别出区块链中的对应区块。与此同时，每个区块都可以通过父区块哈希值字段引用前一区块。通过这样的设计，可以把每个区块链接到各自的父区块，创建了

一条链条，一直可以追溯到第一个区块创世区块。

在 DCB 中，每个区块只能有一个父区块，也只能有一个子区块。区块链的第一个区块称为创世区块。

比特币中的 PoW 由于具备交易速度较慢、消耗算力、浪费公共资源等显性缺陷，因此在 DCB 中平台中结合 PoET 减少算力的损耗。其记录的共识机制被设计成模块化、动态化，可以通过控制链参数进行配置，能动态适用公链和私链的不同应用场景。项目根据数据链本身的应用场景和交易情况，选择合适的共识机制，确保各个分布式节点通过算法取得数据的一致性。四组数组共同组成一个区块。第一组是引用父区块哈希值的数据；第二组是元数据，即难度、时间戳和 Nonce；第三组是元数据的 Merkle 树根；第四组便是动态的数据记录，例如字符串数据，字节流数据等等，根据不同业务场景，可使用不同的记录合约。

交易列表以 Merkle 树表示，包含了产生该区块的所有交易。Merkle 树是一种哈希二叉树，是自底向上构建的。Merkle 树用来归纳一个区块中的所有交易，提供了一种校验区块是否存在某交易的途径。生成一棵完整的 Merkle 树需要递归地对哈希节点进行哈希，并将新生成的哈希节点插入 Merkle 树中，直到只剩一个哈希节点，此节点就是树的根。

4.3 共识机制

如果说共识是区块链的基础，那共识机制就是区块链的

灵魂。共识机制，就是在一个时间段内对事物的前后顺序达成共识的一种算法。在区块链上，每个节点都会有一份记录链上所有交易的账本，链上产生一笔新的交易时，每个节点接收到这个信息的时间是不一样的，有些想要篡改数据的人就有可能在节点上发布一些错误的信息，这时就需要一个人把所有人接收到的信息进行验证，最后公布最正确的信息。

目前比较流行的共识机制有四种：

（1）工作量证明(Proof Of Work，简称 POW)应该算是最常见的共识算法之一，简单理解就是一份用来确认你做过一定量的工作的证明。监测工作的整个过程通常是极为低效的，而通过对工作的结果进行认证来证明完成了相应的工作量，则是一种非常高效的方式。比如现实生活中的毕业证、驾驶证等等，也是通过检验结果的方式（通过相关的考试）所取得的证明。

工作量证明系统（或者说协议、函数），是一种应对拒绝服务攻击和其他服务滥用的经济对策。它要求发起者进行一定量的运算，也就意味着需要消耗计算机一定的时间。这个概念由 Cynthia Dwork 和 Moni Naor 1993 年在学术论文中首次提出。而工作量证明（POW）这个名词，则是在 1999 年 Markus Jakobsson 和 Ari Juels 的文章中才被真正提出。

（2）权益证明机制(Proof of Stake-PoS)也称股权证明，类似于财产储存在银行，这种模式会根据你持有数字货币的

量和时间，分配给你相应的利息。鉴于POW主要依赖于计算机硬件的稀缺性来防止女巫攻击，POS则主要依赖于区块链自身里的代币。在POW中，一个用户可能拿1000美元来买计算机，加入网络来挖矿产生新区块，从而得到奖励。

简单来说，就是一个根据你持有货币的量和时间，给你发利息的一个制度，在股权证明POS模式下，有一个名词叫币龄，每个币每天产生1币龄，比如你持有100个币，总共持有了30天，那么，此时你的币龄就为3000，这个时候，如果你发现了一个POS区块，你的币龄就会被清空为0。

然而，一旦币的权益被用于签名一个区块，则币龄将清为零，这样必须等待至少30日才能签署另一区块。同时，为防止非常老或非常大的权益控制区块链，寻找下一区块的最大概率在90天后达到最大值，这一过程保护了网络，并随着时间逐渐生成新的币而无需消耗大量的计算能力。

权益证明必须采用某种方法定义任意区块链中的下一合法区块，依据账户结余来选择将导致中心化，例如单个首富成员可能会拥有长久的优势。。

（3）拜占庭共识算法（Practical Byzantine Fault Tolerance- PBFT）PBFT 意为实用拜占庭容错算法，该算法由 Miguel Castro(卡斯特罗) 和 Barbara Liskov（利斯科夫）在 1999 年提出来，解决了原始拜占庭容错算法效率不高的问题，将算法复杂度由指数级降低到多项式级，使得拜占庭

容错算法在实际系统应用中变得可行。

PBFT 是一种状态机副本复制算法，即服务作为状态机进行建模，状态机在分布式系统的不同节点进行副本复制。每个状态机的副本都保存了服务的状态，同时也实现了服务的操作。

将所有的副本组成的集合使用大写字母 R 表示，使用 0 到 |R|-1 的整数表示每一个副本。为了描述方便，假设 |R|=3f+1，这里 f 是有可能失效的副本的最大个数。尽管可以存在多于 3f+1 个副本，但是额外的副本除了降低性能之外不能提高可靠性。

（4）消逝时间量证明（时间流逝证明）（Proof of Elapsed Time-PoET）也是一种常见的共识证明。PoET 共识机制算法通常用于许可区块链网络，它可决定网络中获得区块者的挖矿权利。许可区块链网络需要任何预期参与者在加入前验证身份。根据公平彩票系统的原则，每个节点具有同等的可能成为胜出者。PoET 机制赋予大量可能的网络参与者以平等胜出的机会。

PoET 的工作机制如下：网络中的每位参与节点都必须等待一个随机选取的时期，首个完成设定等待时间的节点将获得一个新区块。区块链网络中的每个节点会生成一个随机的等待时间，并休眠一个设定的时间。最先醒来的节点，即具有最短等待时间的节点，唤醒并向区块链提交一个新区块，

然后广播必要的信息到整个对等网络中。同一过程将会重复，以发现下一个区块。

在 PoET 网络共识机制中，需要确保两个重要因素:第一，参与节点在本质上会自然地选取一个随机的时间，而非某一个参与者为胜出而刻意选取了较短的时间;第二，胜出者的确完成了等待时间。

DATACBc 选择 PoW+PoET 作为共识机制，即工作量证明+消逝时间量证明，它的优点是：算法简单，容易实现；节点间无需交换额外的信息即可达成共识；破坏系统需要投入极大的成本，目前基本不可能，而且即使篡改，也无法做到签名数据与以前的一致，用户可以轻松对比发现数据被篡改。

4.4 安全加密算法

本项目采用非对称加密技术，核心加密算法主要是以下几种：

数字签名密钥算法：ECDSA，

椭圆曲线（EC）域参数设定：secp256k（比特币选择的也是该算法），

数字签名签名/验证算法：SHA512withECDSA；

Hash 算法：SHA256、RIPEMD160 算法（RACE 原始完整性校验讯息摘要）；

非对称加密为数据的加密与解密提供了一个非常安全的方法，它使用了一对密钥，公钥（public key）和私钥

（private key）。私钥只能由一方安全保管，不能外泄，而公钥则可以发给任何请求它的人。非对称加密使用这对密钥中的一个进行加密，而解密则需要另一个密钥。

公私钥案例说明

私钥说明：

私钥字节总长度：144 位（后 68 位为公钥的动态字节）

例如私钥字节：

[48, -127, -115, 2, 1, 0, 48, 16, 6, 7, 42, -122, 72, -50, 61, 2, 1, 6, 5, 43, -127, 4, 0, 10, 4, 118, 48, 116, 2, 1, 1, 4, 32, 111, 96, -19, 42, -3, -105, -91, -46, -79, -82, -79, 108, 96, -37, 91, -23, -40, 102, -114, 2, 67, -70, -100, 102, 90, -29, -127, -120, 125, 66, -11, -90, -96, 7, 6, 5, 43, -127, 4, 0, 10, -95, 68, 3, 66, 0, 4, 88, -23, 4, 81, 31, 69, 102, 52, 32, -73, 15, -48, 56, -120, -58, 49, -34, 67, 76, 47, -73, 74, 123, -22, -67, 82, -125, -42, 108, 24, 116, 88, -88, 53, 105, 78, 123, -13, 116, 40, -92, 47, 1, 54, 101, -63, -125, 26, 22, 29, -120, -32, -89, -36, -107, -27, -119, -18, -77, -71, 69, 103, 47, 63]

转换为字符串：

30818d020100301006072a8648ce3d020106052b8104000a047630740201010420 6f60ed2afd97a5d2b1aeb16c60db5be9d8668e0243ba9c665ae381887d42f5a6a00706052b8104000aa144

0342000458e904511f45663420b70fd03888c631de434c2fb74a7beabd5283d66c187458a835694e7bf37428a42f013665c1831a161d88e0a7dc95e589eeb3b945672f3f

公钥说明：

公钥字节总长度：88 位（前 20 位为固定字节，后 68 位为公钥动态字节）

例如公钥字节：

[48, 86, 48, 16, 6, 7, 42, −122, 72, −50, 61, 2, 1, 6, 5, 43, −127, 4, 0, 10, 3, 66, 0, 4, 88, −23, 4, 81, 31, 69, 102, 52, 32, −73, 15, −48, 56, −120, −58, 49, −34, 67, 76, 47, −73, 74, 123, −22, −67, 82, −125, −42, 108, 24, 116, 88, −88, 53, 105, 78, 123, −13, 116, 40, −92, 47, 1, 54, 101, −63, −125, 26, 22, 29, −120, −32, −89, −36, −107, −27, −119, −18, −77, −71, 69, 103, 47, 63]

其中：48, 86, 48, 16, 6, 7, 42, −122, 72, −50, 61, 2, 1, 6, 5, 43, −127, 4, 0, 10 为固定的公钥前缀。为椭圆算法密钥对的算法标识、以及主编码格式标识。

转换为字符串：

3056301006072a8648ce3d020106052b8104000a0342000458e904511f45663420b70fd03888c631de434c2fb74a7beabd5283d66c187458a835694e7bf37428a42f013665c1831a161d88e0a7dc95e589eeb3b945672f3f

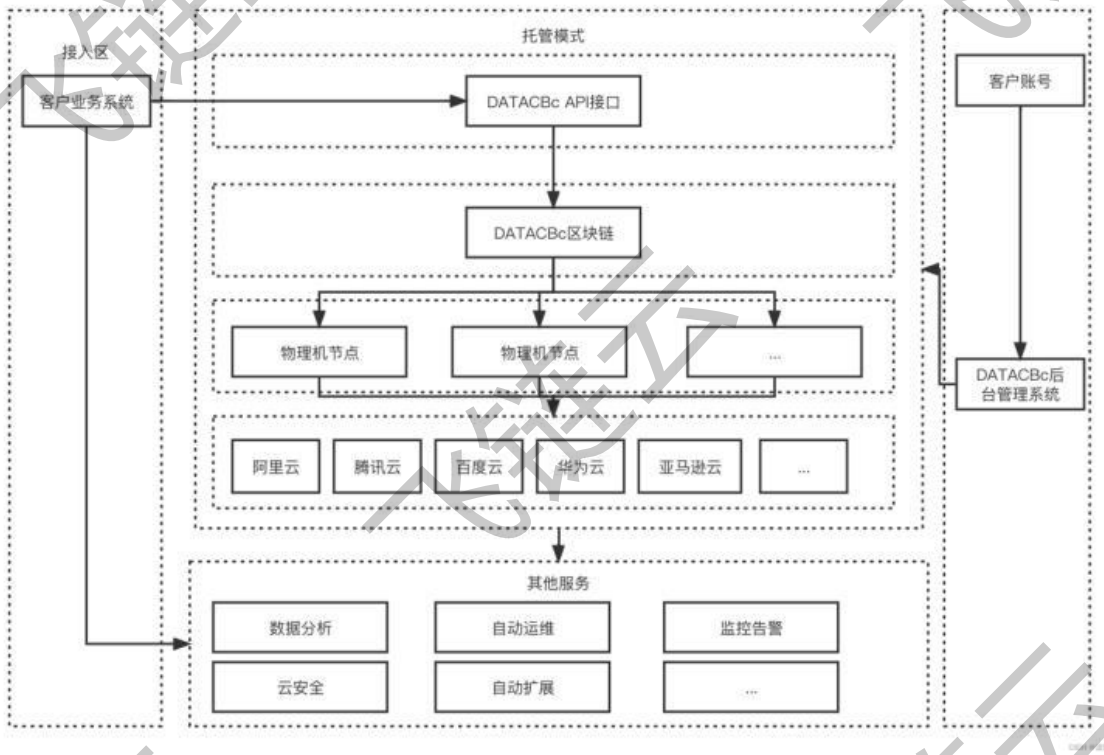
4.5 智能合约

智能合约就是基于密码学技术的数字化合同，是一种计算机程序，而不是传统的纸质合同。智能合约是一段程序，它以计算机指令的方式实现了传统合约的自动化处理。简单讲，智能合约就是双方在区块链资产上交易时，触发执行的一段代码，这段代码就是智能合约。

第五章 Data Copyright Blockchain 相关架构

5.1 基础设施架构

DATACBc 目前一共有两个后端服务。区块链节点认证服务 + 区块链节点服务。



5.2 区块链节点认证服务

功能：区块链节点通过 P2P 通信，传输公钥+签名的数据进行节点认证。

5.3 区块链节点服务

功能：

1、连接区块链节点认证，在节点启动时通过节点公私

钥进行签名，与区块链节点认证服务进行通讯，认证后可正常启动。

2、区块存储结构、数据序列化。

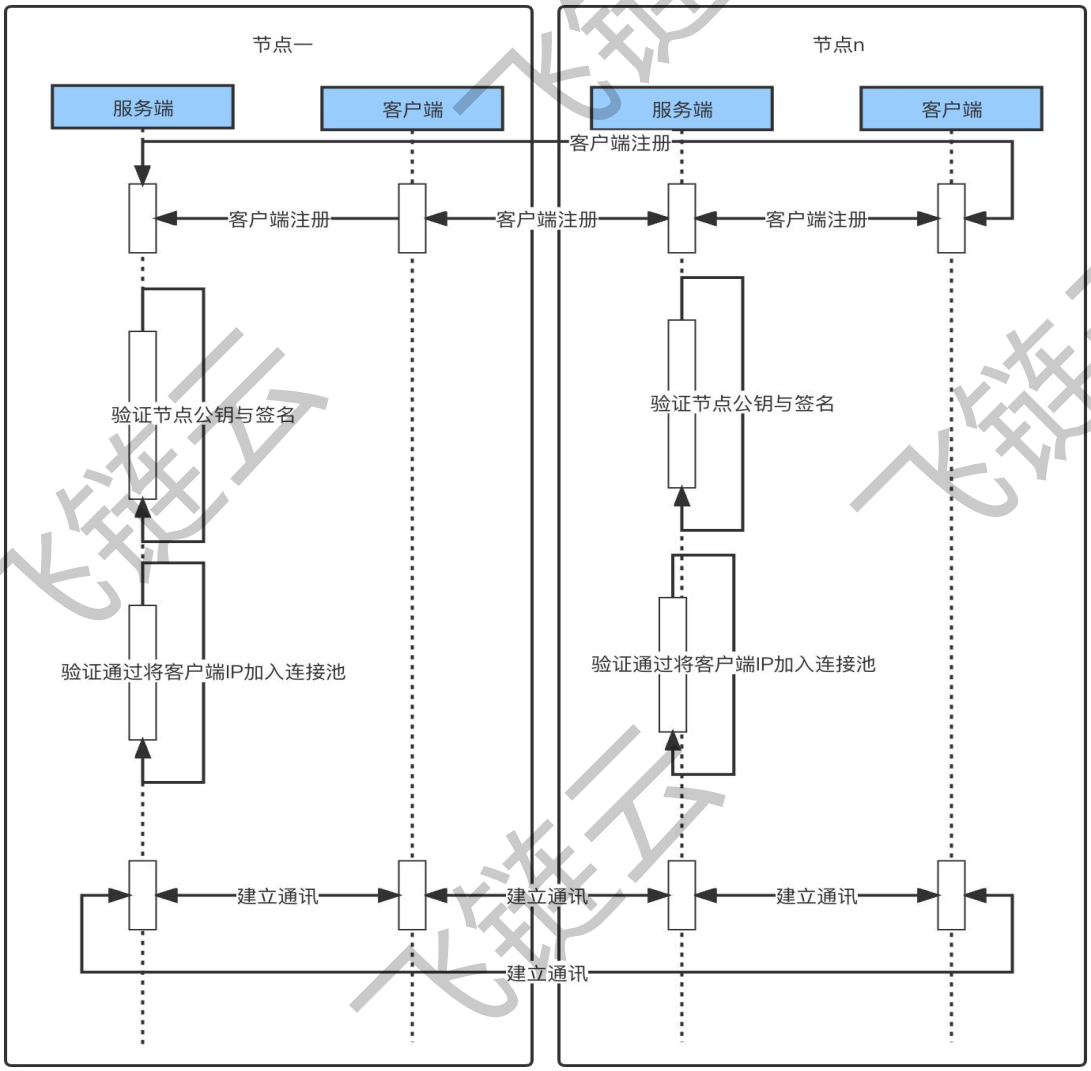
3、交易打包、生成区块链区块。

4、PoW+PoET 共识算法、智能合约。

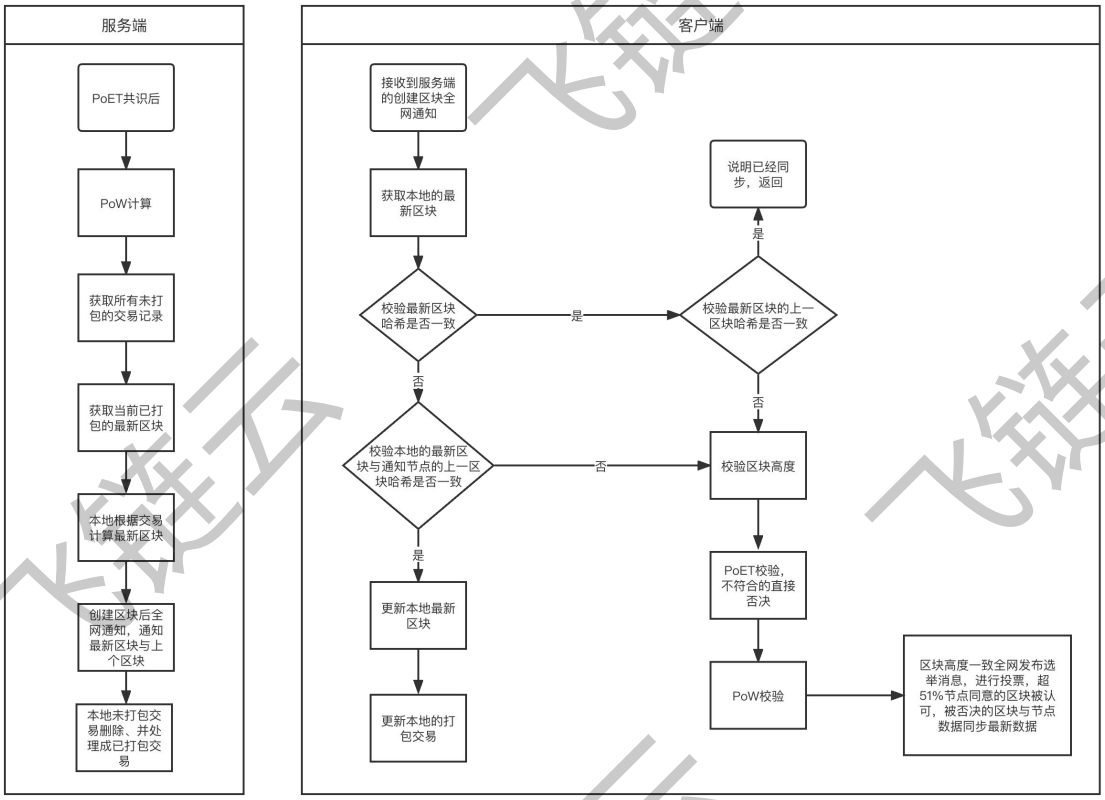
5.4 客户端注册在服务端上流程

每个服务均支持部署在多个服务器上，每个区块链节点均为包含服务端与客户端。不同节点之间通过配置 IP 进行通信。基于 java Websocket 服务。

客户端注册在服务端上流程（单节点&多节点）：



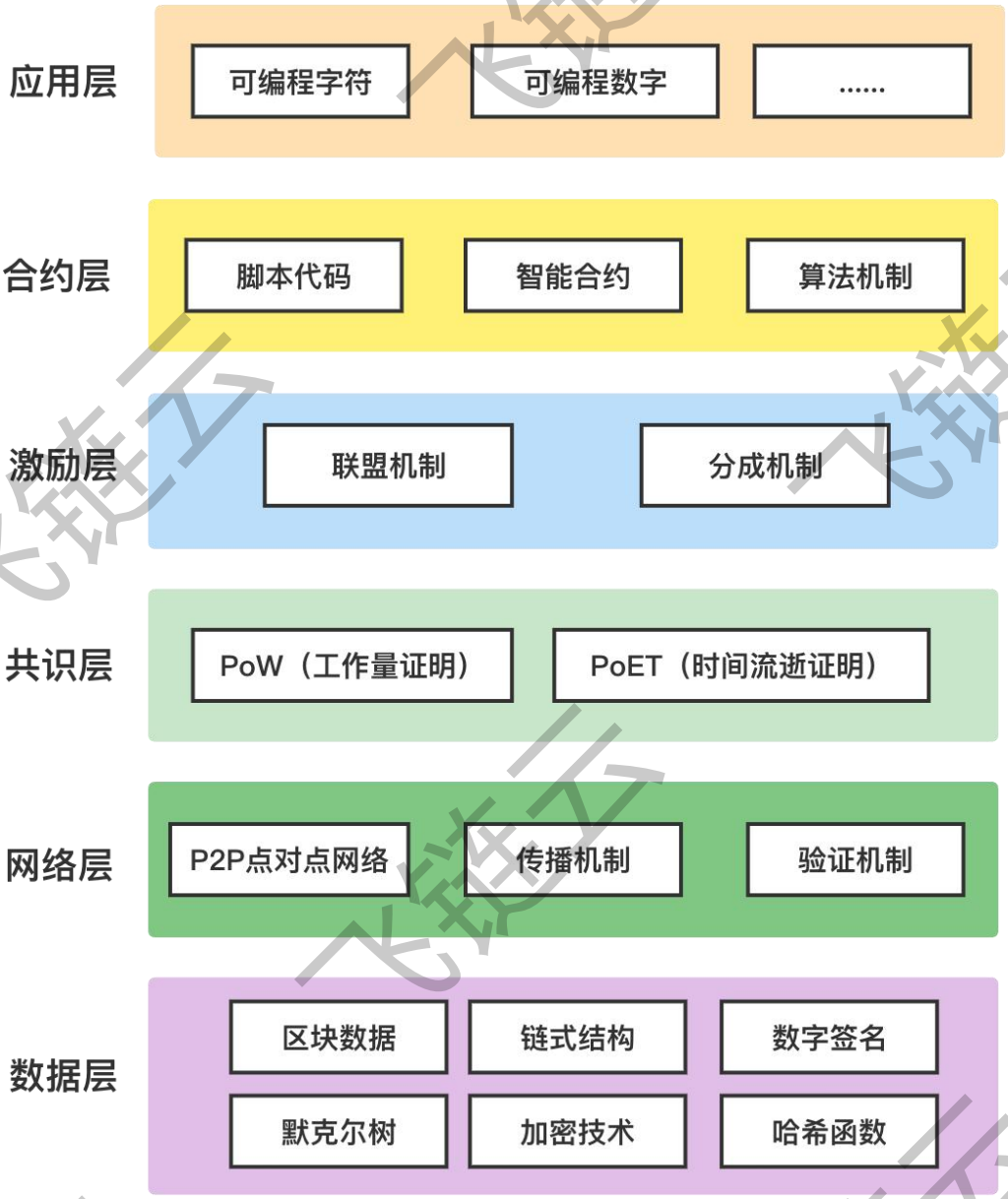
5.5 区块链节点生成区块流程



智能合约校验区块有效性校验项：

区块签名有效性、PoET 共识有效性、PoW 共识有效性、区块高度、默克尔树哈希、区块哈希、业务架构

5.6 区块链架构


应用层
可编程字符
可编程数字
.....
合约层
脚本代码
智能合约
算法机制
激励层
联盟机制
分成机制
共识层
PoW（工作量证明）
PoET（时间流逝证明）
网络层
P2P点对点网络
传播机制
验证机制
数据层
区块数据
链式结构
数字签名
默克尔树
加密技术
哈希函数

第六章 区块链核心页面

区块链节点部署好后,便可以通过访问节点 ip 在浏览中看到区块链浏览器页面。其中有几个比较核心的页面如下:

6.1 区块链浏览器

首页页面的接口地址为:

节点 ip:端口

区块链浏览器首页可以了解到如下的信息:

联盟链标识、当前节点的名称、当前节点的公钥、区块链的区块高度、区块链中累计交易数、最新未打包交易数、该区块链中的节点数

6.2 区块的概要信息

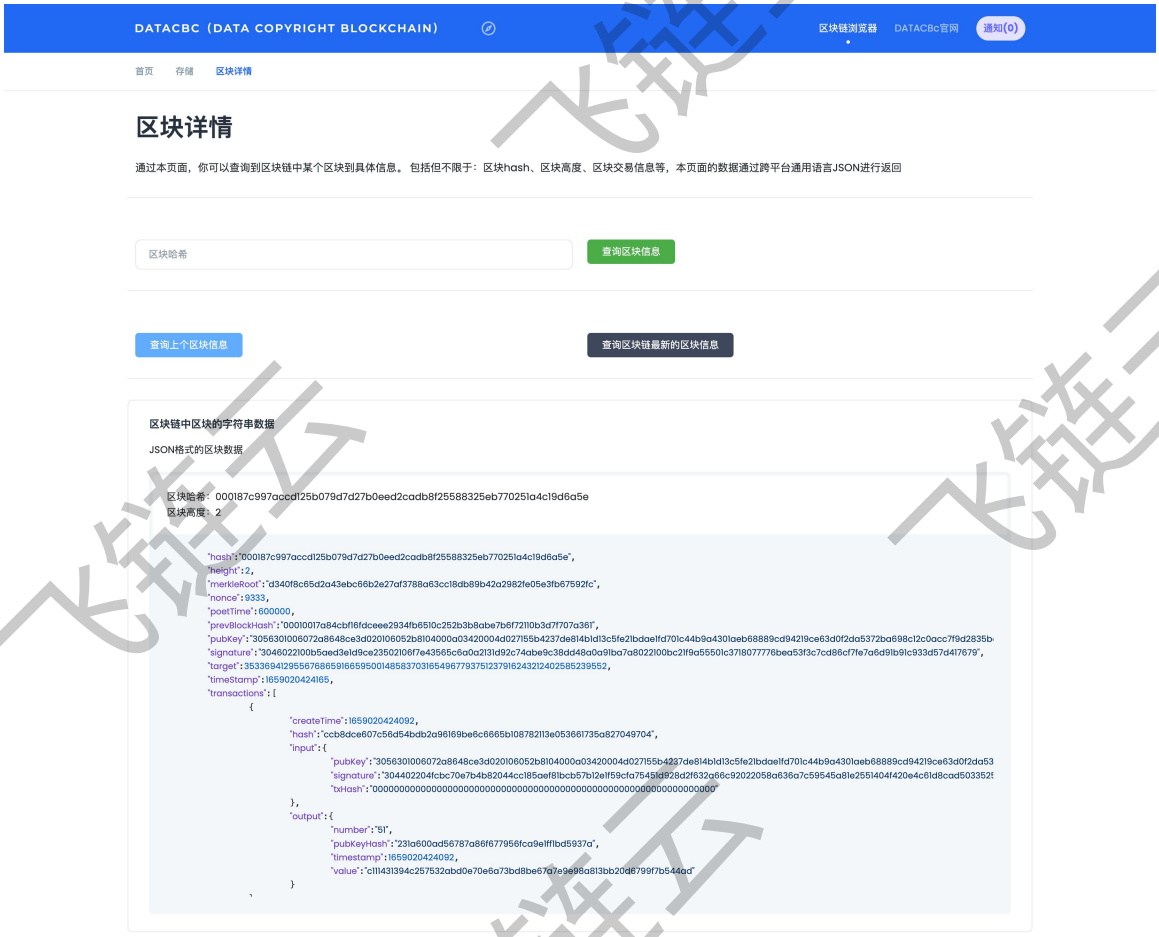
在区块链浏览器首页可以看到最新生成的 20 个区块信

息，表格中包含每个区块的信息如下：

区块高度	区块Hash	区块交易数	交易默克尔树根Hash	上一个区块Hash	工作量证明计数值	区块流逝时间（毫秒）	创建时间	操作
2 0 9 8	0000 84b7 158a 7065 b539 3614 79a8 87a0 ea46 1e34 ba1d 8a13 ac78 fe5e 9328 0b76	1	e9e8 eadd 3839 d700 2fa0 2870 9df1 fca5 bcf0 f7f9 8f0f 8cfe a5fe 5086 72de 9a40	0001 9311 4324 f2dd 5fad 8b1a d39d 1ddf 7f45 b261 8bd1 634f 412f 9f5f b498 7591	6 0 0 8	6 0 0 0 0	2 0 2 2 - 0 8 - 3 0 2 3 : 1 0	详情

6.3 区块详细信息

通过首页点击表格的操作栏目，其中可以点击“详情”，可以查看该区块的详细信息。如下图所示：



在该页面中，用户可以中输入框中输入区块哈希，主动发起查询。也可以通过点击按钮，查询当前区块的上个区块，通过该按钮，可以一直追溯到创世区块。

也可以通过按钮：查询区块链最新的区块信息，一键查询到当前区块链生成的最高高度区块。

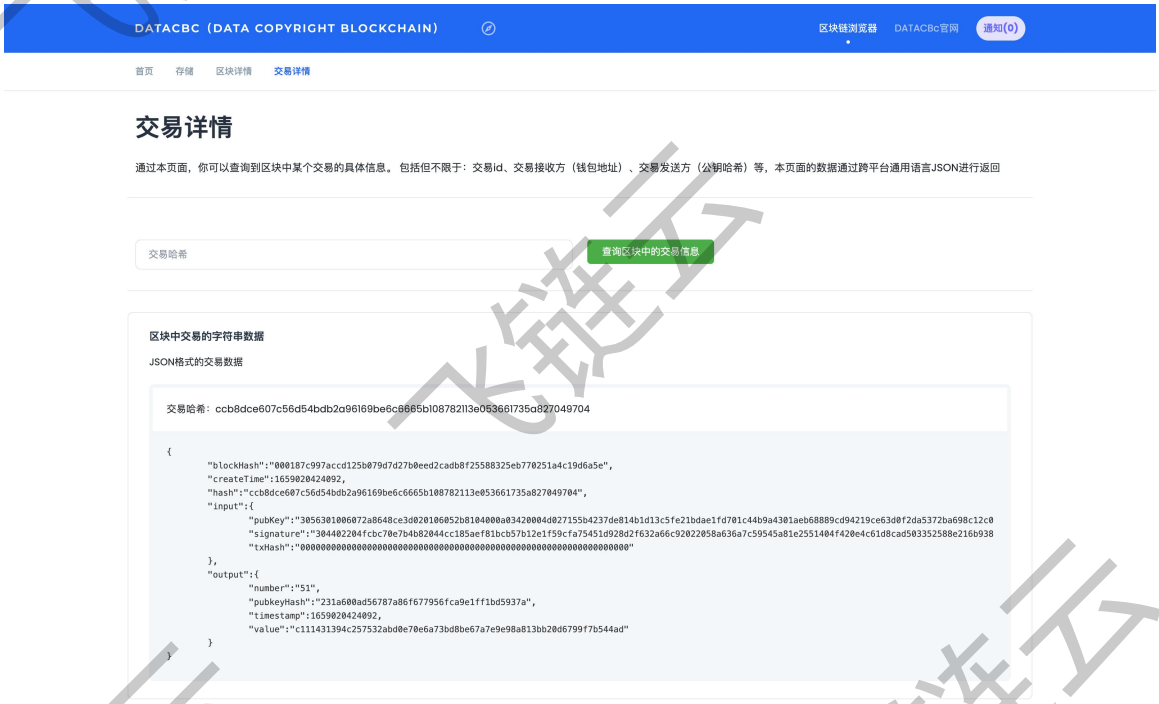
区块信息的结构如下：

{ "hash":"000187c997accd125b079d7d27b0eed2cadb8f25588325eb770251a4c19d6a5e","height":2, "merkleRoot":"d340f8c65d2a43ebc66b2e27af3788a63cc18db89b42a2982fe05e3fb67592fc","nonce":9333, "poetTime":600000,

"prevBlockHash":"00010017a84cbf16fdceee2934fb6510c252b3b8abe7b6f72110b3d7f707a361",
"pubKey":"3056301006072a8648ce3d020106052b8104000a03420004d027155b4237de814b1d13c5fe21bdae1fd701c44b9a4301aeb68889cd94219ce63d0f2da5372ba698c12c0acc7f9d2835bd9a159980588e954177cc056ca9f7",
"signature":"3046022100b5aed3e1d9ce23502106f7e43565c6a0a2131d92c74abe9c38dd48a0a91ba7a8022100bc21f9a55501c3718077776bea53f3c7cd86cf7fe7a6d91b91c933d57d417679",
"target":35336941295567686591665950014858370316549677937512379162432124025852395 52,
"timeStamp":1659020424165,
"transactions":[{ "createTime":1659020424092,
"hash":"ccb8dce607c56d54bdb2a96169be6c6665b108782113e053661735a827049704",
"input":{ "pubKey":"3056301006072a8648ce3d020106052b8104000a03420004d027155b4237de814b1d13c5fe21bdae1fd701c44b9a4301aeb68889cd94219ce63d0f2da5372ba698c12c0acc7f9d2835bd9a159980588e954177cc056ca9f7",
"signature":"304402204fcbc70e7b4b82044cc185aef81bcb57b12e1f59cfa75451d928d2f632a66c92022058a636a7c59545a81e2551404f420e4c61d8cad503352588e216b9384d5007b4",

"txHash":"0000000000000000000000000000000000000000000000000000000000000000" }, "output":{ "number":"51", "pubKeyHash":"231a600ad56787a86f677956fca9e1ff1bd5937a", "timestamp":1659020424092, "value":"c111431394c257532abd0e70e6a73bd8be67a7e9e98a813bb20d6799f7b544ad" } }], "version":"1.0.0" }

6.4 交易详细信息



通过本页面，可以查询到区块中某个交易的具体信息。包括但不限于：交易 id、交易接收方（钱包地址）、交易发送方（公钥哈希）等，本页面的数据通过跨平台通用语言 JSON 进行返回。

单个交易数据的结构示例如下所示：

{ "blockHash":"000187c997accd125b079d7d27b0eed2cadb8f25588325eb770251a4c19d6a5e",
"createTime":1659020424092,
"hash":"ccb8dce607c56d54bdb2a96169be6c6665b108782113e053661735a827049704",
"input":{ "pubKey":"3056301006072a8648ce3d020106052b8104000a03420004d027155b4237de814b1d13c5fe21bdae1fd701c44b9a4301aeb68889cd94219ce63d0f2da5372ba698c12c0acc7f9d2835bd9a159980588e954177cc056ca9f7",
"signature":"304402204fcbc70e7b4b82044cc185aef81bcb57b12e1f59cfa75451d928d2f632a66c92022058a636a7c59545a81e2551404f420e4c61d8cad503352588e216b9384d5007b4",
"txHash":"0000000000000000000000000000000000000000000000000000000000000000" }, "output":{ "number":"51",
"pubkeyHash":"231a600ad56787a86f677956fca9e1ff1bd5937a",
"timestamp":1659020424092,
"value":"c111431394c257532abd0e70e6a73bd8be67a7e9e98a813bb20d6799f7b544ad" } }

6.5 节点的存储信息与最新区块信息

该区块链浏览器页面主要展示整个区块链的概览信息、区块链的存储信息和最新的区块信息。

节点区块链的大小，交易数量，最新区块的哈希、高度等均可以在该区块链浏览器页面进行查看。

6.6 接口 API

为了方便开发者接入，每个节点均提供了公共的 API 接口，可以由开发者自行创建公私钥，自行进行签名交易，查询区块，查询交易，查询藏品等操作

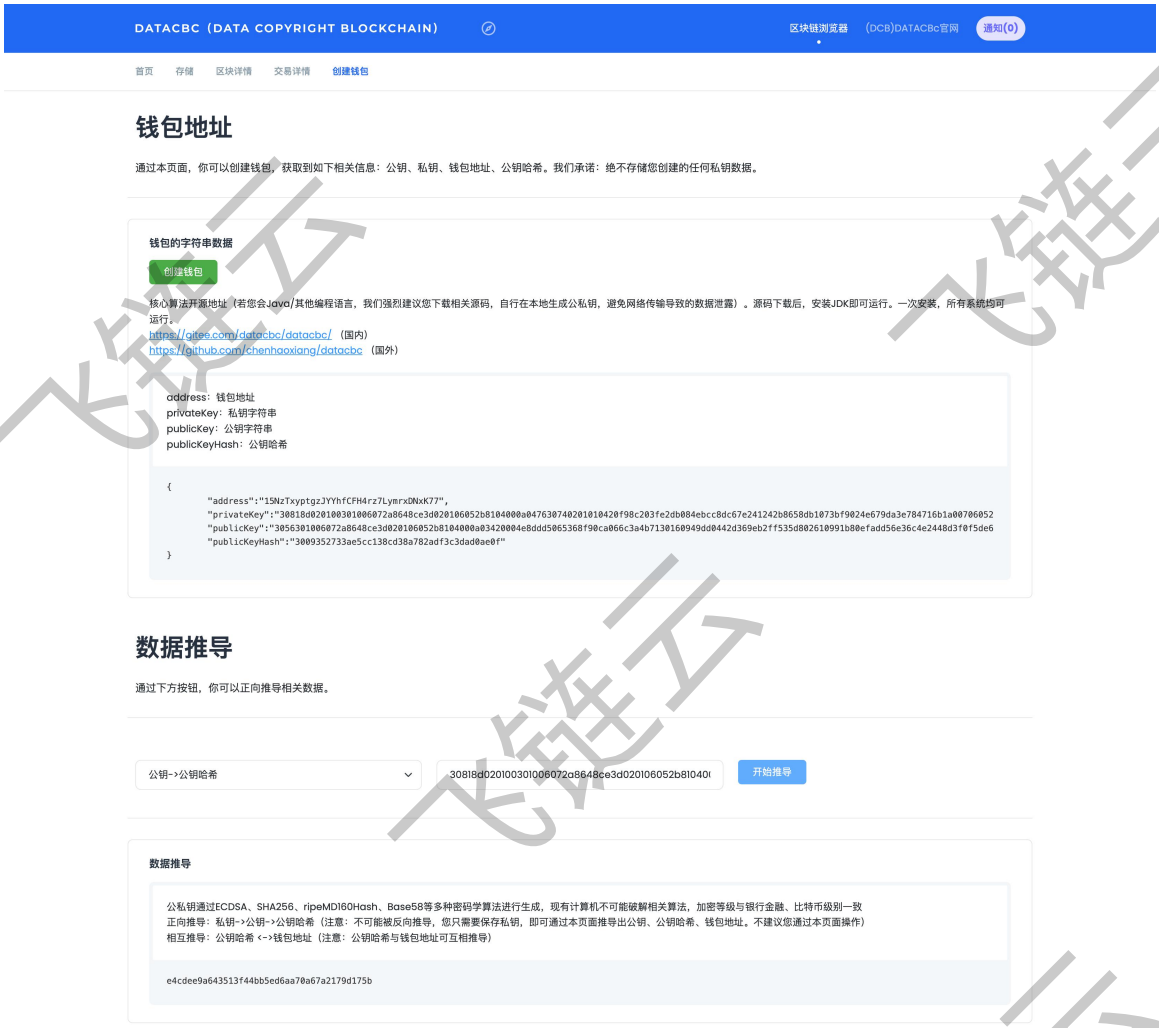
页面的接口地址为：

节点 ip:端口/swagger-ui/index.html

展示页面如下图所示：

6.7 区块链浏览器钱包

地址：节点 ip:端口/create-wallet



通过本页面，你可以创建钱包，获取到如下相关信息：公钥、私钥、钱包地址、公钥哈希。我们承诺：绝不存储您创建的任何私钥数据。

公私钥通过 ECDSA、SHA256、ripeMD160Hash、Base58 等多种密码学算法进行生成，现有计算机不可能破解相关算法，加密等级与银行金融、比特币级别一致

正向推导：私钥->公钥->公钥哈希（注意：不可能被反向推导，您只需要保存私钥，即可通过本页面推导出公钥、公钥哈希、钱包地址。不建议您通过本页面操作）

相互推导：公钥哈希 <->钱包地址（注意：公钥哈希与钱包地址可互相推导）

若您会 Java/其他编程语言，我们强烈建议您下载相关源码，自行在本地生成公私钥，避免网络传输导致的数据泄露。源码下载后，安装 JDK 即可运行。一次安装，所有系统均可运行。

第七章 区块基础数据结构

DATACBc 可以视为一个数据库，数据库数据的变更由记录/交易上链催化。为了有效、有序管理记录，必须将一笔或多笔记录组成一个数据块，才能提交到数据库中。这个数据块即区块(Block)。一个区块不但包含了多笔记录，还记录一些额外数据，以便正确提交到数据库中。

7.1 高度为 1756 的区块结构

创世区块由于没有上个区块，上一个哈希字符串固定为 64 位 0 的字符串；

下面是一份 JSON 格式的区块结构示例：

其中 transactions 部分数据为记录合约动态内容，本部分内容为数字藏品记录合约使用。

{"hash":
"0001786f3505b3c3fd91fed7d905eff54d2d4af1524c9fd80168
a4c48cd5032d",
"height": 1756,
"merkleRoot":
"183ef6ce1b90171b5341240a390d0693110f089255b040bbaae1
8396d51eddc3",
"nonce": 38081,
"poetTime": 600000,

```
        "prevBlockHash":
"0001c23f466c4a1b98ef594aa9267d73f73947dec1f72a58a6e4
5e6328ba433d",
        "pubKey":
"3056301006072a8648ce3d020106052b8104000a03420004da3c
8fac604265f13ab48e11cf5a842d7799ef6bb913adb78290a2c0
2508ddc3c03b094f6edc2a52151f725fd2e47835a437f45cee15
86d5a872862768a23b25",
        "signature":
"3045022100e1a038e8b49d83b092be89baa226eb09582706e38
f58c6d6707d932e437472b5022029056e072ee6ec46e5d7ca851
490118a52e23c6acaede282915c9d241295cdae",
        "target":
353369412955676865916659500148583703165496779375123
7916243212402585239552,
        "timeStamp": 1661303658010,
        "transactions": [
            {
                "createTime": 1661303270851,
                "hash":
"207bc9b7e2b30dad978ff4898c6e79e31b83d45ef26289bfe47
f6e3b048f64ff",
```

```
            "input": {
                "pubKey":
"3056301006072a8648ce3d020106052b8104000a034200049a65
7d6d76c608369f217f516fe5a1193d1bbaf7f2413c1598fbc6fd1
eff20efead5c5bdb249771df94cc622d858ab542d09d10b52938
df85813bddc988f58db",
                "signature":
"304402206da33f9109cf6f30d7bb00b10902e4fd3b65305bea49
cb685db1a0560eb47069022078984ec764b75bc94afe9c3cb5ed
ba1c2ca0bbdbdc39d5aab0f805fb55faaa93",
                "txHash":
"a5f3a2f4ceb4cd97c667031a2530144196e5cf2cff026bcb2f0a
e86b1f8eccab"
            },
            "output": {
                "number": "136",
                "timestamp": 1661303270851,
                "value":
"ebbb4c9e97f799121df7ae38fd471aa22d6e229e027ac27a81ca
779e6b54dd0a"
            }
        },
```

```
{
    "createTime": 1661303582902,
    "hash":
"c8e9450a9db14afa0b280c889e8b3754d052efbaa1fed985f230
a739b73dab50",
    "input": {
        "pubKey":
"3056301006072a8648ce3d020106052b8104000a0342000497a5
40528baed3dd49c2107fcee48fb92cecd9404a2d193a25a99ead
908c9c8df69d6b88e0a53a1686c559460aeb2bde41e4f3db7e77
d8486866138b22b3ba80",
        "signature":
"304402203131a24f8a601e733063cb64d870d9c951f2b72388a
54fbade81c5197079b03f022002a99c60549bd9da9ff470cb452c
cd06d0beaff980cf4de4ac1591753295ff9d",
        "txHash":
"23624fa05b3a83ab608b049aed31f9a5fd56e6be2eedb099a4fd
cdcdb9119ce7"
    },
    "output": {
        "number": "421",
        "timestamp": 1661303582902,
```

"value":
"3ead8ca4bf8bc591c7f0d3f3a09747674d9a63a94dfb7cdcc9ee419b348242d9"
}
}
],
"version": "1.0.0"}

区块分为两部分：区块头(Header)和区块体(Body)。区块头信息量非常丰富，不但和上一个单元建立联系还记录了一些记录执行情况信息和签名信息等。其中非常重要的概念Trie，全名是默克尔压缩前缀树，其为检索和校验记录带来了指数级运算等提升。

其中 Header json key 的描述可以看下面注释：

```
/**
 * 区块链版本号，根据不同的版本，会有不同的处理。解决 bug 或者升级区块链
 */
private String version = "1.0.0";
/**
 * 当前区块的 hash 值，用于校验区块是否正确挖出。
 */
private String hash;
```

```
/**
 * 签名,联盟节点的签名 hash 的签名
 */
private String signature;
/**
 * 签名,联盟节点的公钥
 */
private String pubKey;
/**
 * 前一个区块的哈希值，记录此区块直接引用的父区块哈希值。通过此记录，才能完整的将区块有序组织，形成一条区块链。并且可以防止父区块内容被修改，因为数据修改，区块哈希必然发生变化，因此一个区块直接或间接的强化了所有父辈区块，通过加密算法保证历史区块不可能被修改。
 */
private String prevBlockHash;
/**
 * 交易信息
 */
private Collection<Transaction> transactions;
/**
```

```
     * 区块创建时间(单位:毫秒)
     */
    private Long timeStamp;
    /**
     * 时间流逝证明（单位：毫秒）
     * 下一个区块的创建时间 》 上个区块创建时间
+poetTime
     */
    private Long poetTime;
    /**
     * 工作量证明计数器 挖到时的运算值
     */private BigInteger nonce;/**
     * 表示此区块高度。用于对区块标注序号，在一条区
块链上，区块高度必须是连续递增。
     */
    private Integer height;
    /**
     * 默克尔树 rootHash
     * 一个哈希值，表示执行完此区块中的所有记录/交易
后 DATACBc 状态快照 ID。因为 DATACBc 描述为一个状态机系
统，因此快照 ID 称之为状态哈希值。又因为状态哈希是由所
有账户状态按默克尔前缀树算法生成，因此称为状态默克尔
```

树根值。

```
     */
    private String merkleRoot;
    /**
     * 难度目标值
     * 表示此区块能被挖出的难度系数
     */
    private BigInteger target;
```

其中 Body 主要为记录合约的动态结构数据。在上面结构中，transactions 字段为记录合约字段。DATACBc 底层的记录合约是可以动态增减的，其通过 JSON 格式字符串定义了相关元数据类型。其元数据类型目前包含：

整型字段

字符串型字段

二进制字段

数字字段

元数据类型结构：

```
{"key": "name-字段名称",
  "describe": "描述",
  "value": "业务值",
  "type": "字段类型，整型、字符串等"}
```

7.2 记录合约分为五大类

根据记录合约的不同，可以动态配置出不同的链用途。现可支持的记录合约分为五大类：

数字版权类记录合约

司法取证、溯源、记录类合约

虚拟货币、数字交易记录类合约

数据二进制上链类合约

其他自定义合约

创建区块关键 Java 代码：

```
/**
 * 创建区块
 * @param lastBlock 上个区块
 * @return
 */private synchronized static Block createBlock(Block lastBlock,String publicKey,String privateKey,long poetTime,String product) {
        //获取交易
        List<Transaction> transactions = RocksDBNoPackageTransactionUtils.getInstance(product).clearAndGetTransactionList();
        if(CollectionUtils.isEmpty(transactions)){
            throw new ParameterExpression("不存在交易，
```

```
不进行生成区块");
        }
        Block genesisBlock = null;
        try {
            if (lastBlock == null) {
                //不存在就创建创世区块
                genesisBlock =
Block.newGenesisBlock(transactions,poetTime,publicKey,pri
vateKey);
            }else {
                if(     System.currentTimeMillis()     <=
(lastBlock.getTimeStamp() + poetTime) ){
                    //立刻进行生成区块,至少需要 10 分
钟
                    log.info("从上个区块生成开始，时间
尚未流逝完 ，当前时间：{},上个区块创建时间：{},需要流逝
时
间 :{}",System.currentTimeMillis(),lastBlock.getTimeStamp(),
poetTime);
                    throw new ParameterExpression("从
上个区块生成开始，时间尚未流逝完");
                }
```

```
                genesisBlock =
Block.newBlock(lastBlock.getHash(), transactions,
lastBlock.getHeight() + 1, poetTime,publicKey,privateKey);
            }
            //保存区块

RocksDBUtils.getInstance(product).putBlock(genesisBlock,pr
oduct);
            //保存交易

RocksDBCacheTransactionUtils.getInstance(product).putTra
nsactions(transactions,product);
        } catch (Exception e) {
            log.error("生成新的区块出现异常，进行回滚，
上个区块:"+ JSON.toJSONString(lastBlock),e);
            //数据进行回滚,清空本地的区块,恢复本节点
的交易.每次后一步均会处理本节点前一步的原子性操作

RocksDBNoPackageTransactionUtils.getInstance(product).pu
tTransactions(transactions);

RocksDBUtils.getInstance(product).deleteBlock(genesisBloc
```

k,lastBlock,product);

RocksDBCacheTransactionUtils.getInstance(product).delete Transactions(transactions,product);

}

return genesisBlock;}

工作量证明运行核心代码：

```
/**
 * <p> 创建新区块 </p>
 * 在这里会进行工作量证明
 * @param previousHash
 * @param transactions
 * @return
 */public static Block newBlock(String previousHash, Collection<Transaction> transactions, int height,long poetTime,String pubKey,String privateKey) {
        Block block = new Block(poetTime,pubKey);
        block.setPrevBlockHash(previousHash);
        block.setTimeStamp(System.currentTimeMillis());
        block.setHeight(height);
        block.setTransactions(transactions);
```

```
block.setMerkleRoot(ByteUtils.bytesToHexString(block.hash
Transaction()));

        ProofOfWork pow =
ProofOfWork.newProofOfWork(block);
        //计算
        PowResult powResult = pow.run();
        block.setHash(powResult.getHash());
        block.setNonce(powResult.getNonce());
        //对 hash 进行签名
        try {
            byte[] signBytes =
ECDSAUtils.sign(ByteUtils.hexStringToByte(block.getHash()),
ByteUtils.hexStringToByte(privateKey));

block.setSignature(ByteUtils.bytesToHexString(signBytes))
;
        } catch (Exception e) {
            log.error("签名异常",e);
            throw new ServiceException("签名异常，请检
查私钥");
        }
```

```
        log.info("创建新的区块：{}",
JSON.toJSONString(block));
        return block;}
```

工作量证明运算核心代码：

```
    /**
     * 创建新的工作量证明，设定难度目标值
     * <p>
     * 对1进行移位运算,将1向左移动 (256 - difficulty)
位，得到我们的难度目标值
     *
     * @param block
     * @return
     */public static ProofOfWork newProofOfWork(Block
block) {
        BigInteger targetValue =
TARGET_MAX.divide(difficulty);
        return new ProofOfWork(block, targetValue);}
    /**
     * 运行工作量证明，开始挖矿，找到小于难度目标值
的 Hash
     *
     * @return
```

```
*/public PowResult run() {
    BigInteger nonce = new BigInteger("0");
    String shaHex = "";
    this.block.setTarget(target);
    long startTime = System.currentTimeMillis();
    while (nonce.compareTo(TARGET_MAX) < 0) {
        this.block.setNonce(nonce);
        byte[] data = this.prepareData(nonce);
        shaHex = DigestUtils.sha256Hex(data);
        if (new BigInteger(shaHex,
16).compareTo(this.target) < 0) {
            log.info("计算区块完成，正确的 Hash 值:
{}，共计算：{}次,花费时间:{}ms", shaHex, nonce.longValue(),
(System.currentTimeMillis() - startTime));
            break;
        } else {
            nonce = nonce.add(new BigInteger("1"));
        }
    }
    return new PowResult(nonce, shaHex, target);}
```

第八章 团队介绍

DATACBc 团队成员有热血、有斗志、有梦想、有能力，他们坚持诚信、专业、专心、专注的态度把全部精力投入在 DATACBc 的发展。项目主要负责人有着 10 余年的 Java 行业经验，8 年的区块链技术经验，拥有 Java 领域相关专利 8 余项，SaaS 领域相关专利 6 余项（含终审&下发），数本 Java 领域、数字藏品领域、SaaS 低代码领域书籍的出版（含已出版&与出版社签订合同）；

团队人员擅于理解客户的需求，把需求转化为产品。在项目管理方面也有着丰富的经验，承接过多个国内外大型项目。团队成员以强大的技术实力和丰富的行业经验全力支撑项目的成长和发展。

第九章 交易（记录）数据基础数据结构

交易(Transaction)是指由一个外部账户转移一定资产给某个账户， 或者发出一个消息指令到某个智能合约。

在 DATACBc 中，支持仅存储数据的智能记录合约，而不需要输出方，此时不称该 Transaction 为交易，而称之为记录。在本文档中，您基本可以把记录和交易当作一个类型看待，仅仅是结构数据的不同。

在 DATACBc 网络中，交易/记录执行属于一个事务。具有原子性、一致性、隔离性、持久性特点。

原子性： 是不可分割的最小执行单位，要么做，要么不做。

一致性： 同一笔交易执行，必然是将 DATACBc 账本从一个一致性状态变到另一个一致性状态。

隔离性： 交易执行途中不会受其他交易干扰。

持久性： 一旦交易打包，则对 DATACBc 账本的改变是永久性的。后续的操作不会对其有任何影响。

因为是事务型，因此我们需确保在执行事务前让交易符合一些设计要求。

交易必须唯一，能区分不同交易且同一笔交易不能重复提交到账本中。

交易内容不得变化，每个节点收到的交易都必须一致，

交易执行时账本状态变化也是一致的。

交易必须被合法签名，只有已正确签名的交易才能被执行。

交易不能占用过多系统资源，影响其他交易执行。

对交易的设计要求，涉及软件系统的方方面面，但最基础部分还是交易数据本身。

DATACBc 暂不支持单条链中使用多个不同的智能记录合约，但是由于使用的算法合约等一致，记录本身在不同链可以互通。

第十章 序列化方式与存储

出于智能合约兼容、并发性能和安全方面的考虑，我们选择了在 Java 序列化中非常快的两种序列化方式：

10.1 fastjson + Kroy

DATACBc 是 BaaS 区块链服务，为了兼容后续智能合约的变化，我们引入了 json 序列化方式，可以方便兼容不同的智能合约。由于 fastjson 出过好几次安全性事件，综合考虑下，我们仅仅中合约兼容中使用了最简单的 json 序列化，而最终的存储数据我们会将 json 字符串进行 Kroy 序列化，避免由于 fastjson 序列化引起的安全事件。

出于便捷部署和存储性能考虑，我们选择了非常适合分布式存储的数据库：RocksDB

10.2 fastjson 介绍

fastjson 是阿里巴巴的开源 JSON 解析库，它可以解析 JSON 格式的字符串，支持将 Java Bean 序列化为 JSON 字符串，也可以从 JSON 字符串反序列化到 JavaBean。

fastjson 相对其他 JSON 库的特点是快，从 2011 年 fastjson 发布 1.1.x 版本之后，其性能从未被其他 Java 实现的 JSON 库超越。

fastjson 在阿里巴巴大规模使用，在数万台服务器上部署，fastjson 在业界被广泛接受。在 2012 年被开源中国评选为最

受欢迎的国产开源软件之一。

开源地址：https://github.com/alibaba/fastjson

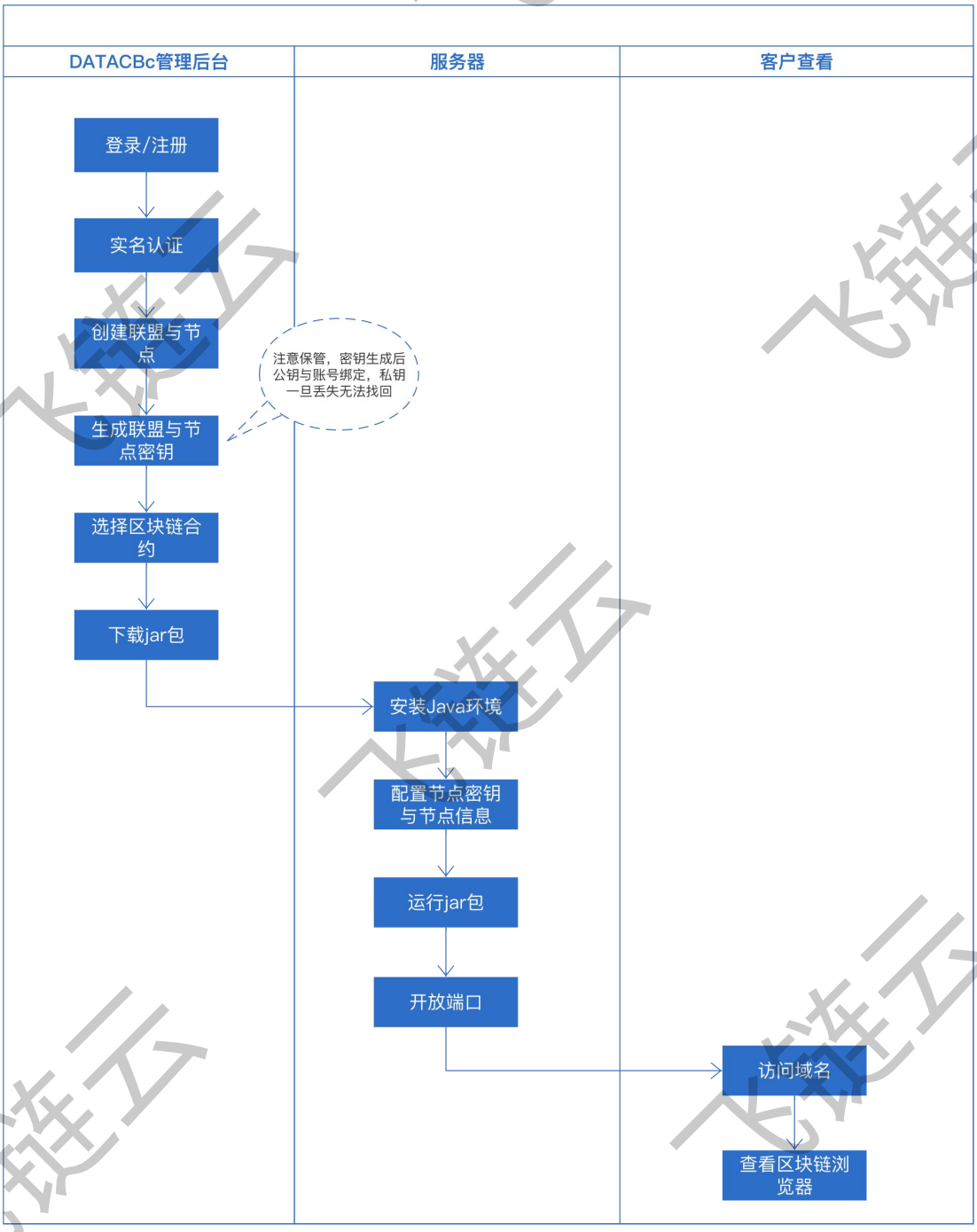
10.3 Kroy 介绍

Kroy 序列化，号称 Java 中最快的序列化框架。Kroy 是一个快速序列化/反序列化工具，依赖于字节码生成机制（底层使用了 ASM 库），因此在序列化速度上有一定的优势。

10.4 RocksDB 介绍

RocksDB 是用于键值数据的高性能嵌入式数据库。它是 Google 的 LevelDB 的一个分支，经过优化，可以利用许多 CPU 内核，并有效利用快速存储（例如固态驱动器）来处理输入/输出受限的工作负载。它基于日志结构的合并树数据结构。它是用 C ++编写的，并为 C ++，C 和 Java 提供了正式的语言绑定。

第十一章 DATACBc 使用步骤



每增加一个节点会生成一个节点的公私钥（密钥用于节点认证、区块与交易的生成签名；私钥请务必保管好，DATACBc 不会存储任何用户的私钥）。

若私钥不慎丢失或者泄露，只能在DATACBc管理后台进行重新生成。泄露节点与其他节点需要重新配置新的公钥，泄露的公私钥即作废。

第十二章 风险提示

（1）司法监管相关的风险

区块链技术已经成为世界上各个主要国家监管的主要对象，如果监管主体插手或施加影响，则应用或代币可能受到其影响。例如法令限制使用、销售电子代币等，代币有可能受到限制、阻碍甚至终止应用的发展。

（2）应用缺少关注度的风险

平台应用存在没有被大量个人或组织使用的可能性，这意味着公众没有足够的兴趣去开发和发展这些相关分布式应用，这样一种缺少兴趣的现象可能对代币和应用造成负面的影响。

（3）相关应用或产品达不到预期标准的风险

平台自身在开发阶段，在发布正式版之前可能会进行比较大的改动，或是在发布之前市场经历巨大的变革，造成平台在功能或是技术上未达到预期要求。或是因为错误的分析，导致平台的应用或是代币的功能未能达到预期。

（4）破解的风险

目前使用的技术无法被破解，但假设密码学飞速发展，或是计算机运算速度飞速进步，诸如量子计算机的发展，或将带来破解的风险，导致代币的丢失。

（5）其他说明

请充分了解运营平台的发展规划以及清楚区块链行业的相关风险，否则不建议参与本次投资，如果您进行投资，代表您确认已经完全理解并认可细则中的各项条款说明。

第十三章 免责声明

本文档只用于传达信息之用途，并不构成本项目买卖的相关意见。以上信息或分析不构成投资决策权参考依据。本文档不构成任何投资建议、投资意向或教唆投资。

本文档不组成也不理解为提供任何买卖的行为，也不是任何形式上的合约或是承诺。

相关意向用户需明确了解本项目的风险，投资者一旦参与投资即表示了解并接受该项目风险，并愿意个人为此承担一切相应结果或后果。

运营团队不承担任何参与本项目并因本项目造成的直接或间接的损失。

第十四章 相关网址

DATACBc 核心算法开源地址

https://gitee.com/datacbc/datacbc/ （国内）

https://github.com/chenhaoxiang/datacbc （国外）

DCB 官网地址

https://datacbc.com

飞链云部分产品地址

https://feilianyun.cn

https://nft.feilianyun.cn

https://ai.feilianyun.cn