EOTC

全球首个去中心化 OTC 交易所

白 皮 书

出于保密原因，本白皮书并未说明 EOTC 全部事项。要完全理解本白皮书的目的、状态和限制条件，请务必先阅读文件末尾的免责和风险声明部分。

目 录

1. 简介

在过去的十年中，中心化交易所已经暴露出大量的安全与作恶漏洞,。一方面，由于不受监管，容易产生交易所监守自盗和市场操控的风险。另一方面，中心化交易所面临越来越大的监管压力，随着中国大陆最严厉的“924 通知”出台，火币、币安等中心化加密货币交易所纷纷宣布退出中国大陆市场。从种种迹象看来，中心化交易所的前景堪忧。

在这一背景下，新一代去中心化 OTC 交易所 EOTC 诞生。EOTC 是全球第一个去中心化 OTC 交易所。与常见的中心化交易所和去中心化币币交易所不同，EOTC 采用的是完全去中心化去信任化的方式，支持各种法币与区块链代币的兑换。我们将为用户实现：

- 一个公开透明的智能数字货币交易环境和资产管理服务。
- 安全、稳定的资产账户，彻底清理黑钱，还币圈一片净土。
- 主流法币和区块链数字货币的兑换服务。

EOTC 的愿景是在创建不受任何中心化组织机构控制的高效、公平与无须信任的去中心化 OTC 交易所，让用户在完全自主掌握个人资产的前提下进行高效科学的投资。并通过 EOTC 底层技术实现全球化数字金融互通网络，实现金融自由、网际自由。

未来 EOTC 将不断发展项目生态产品体系，在原有已形成的币币交易、法币交易、杠杆交易的同时，不断扩大生态的覆盖面积，未来业务范围将覆盖理财、ZAPP、万人热聊社交端、支持多币种红包、物联网设备落地、元宇宙等多种产品线于一体的丰富生态，构建强大的 EOTC 生态王国。

2.项目背景

2.1 去中心化金融

金融行业是一个古老的行业，但绝不是一个落后的行业，相反，金融一定是一个与时俱进的行业。正是基于此，科技才是当代金融发展的重要驱动因素，可以和制度与实践并列为三大支柱。任何通过制度的限制以及对创新的约束，都与这个行业最本质的初衷背道而驰。所以金融一定是一个具有科技基因的行业。金融的发展最后必然指向普惠金融，数字科技本身就是在这个基础上扩大服务覆盖面。市场蓬勃发展之后，我们所需要解决的重大问题之一，就是如何更有效的监管数字金融，防范可能产生的风险的社会性、外溢性以及极具杀伤力的破坏性。

加密数字货币的出现，显示数字化对金融产业链、金融组织的边界带来了重构的可能性。各国政府对数字货币的态度虽然不尽相同，但总体趋势上来看，传统金融业对于加密数字货币的接受程度越来越高：很多金融组织或者大公司都推出了加密数字货币，一些国家的央行也正在考虑推出自己的数字货币。但是，作为一种新生事物，数字货币的金融基础设施的缺乏已经开始倒逼市场的变革。

中心化金融存在的种种问题或风险，有行业垄断程度加剧的原因，也有从业机构内部管理的原因。在传统的信息技术条件下，人和人之间的组织关系是无法改变的。而金融业恰恰是一个极其依赖人际关系的行业，其发展到极致就必然会组织结构提出新的要求，传统组织结构的中心化结构带来的低效和信息不透明则是最大的阻碍。区块链技术正是传统价值分配体系的矛盾深化，无法对生产力提供更多支持而带来的客观求变的结果。

中心化金融存在的种种问题或风险，有行业垄断程度加剧的原因，也有从业机构内部管

理的原因。在传统的信息技术条件下，人和人之间的组织关系是无法改变的。而金融业恰恰是一个极其依赖人际关系的行业，其发展到极致就必然会组织结构提出新的要求，传统组织结构的中心化结构带来的低效和信息不透明则是最大的阻碍。区块链技术正是传统价值分配体系的矛盾深化，无法对生产力提供更多支持而带来的客观求变的结果。

2018 年，DeFi 概念诞生，DeFi 全称为 Decentralized Finance，即“去中心化金融”或者“分布式金融”。“去中心化金融”与传统中心化金融相对，指建立在开放的去中心化网络中的各类金融领域的应用，目标是建立一个多层面的金融系统，以区块链技术和密码货币为基础，重新创造并完善已有的金融体系。

表 1-1：传统金融/Fintech/DeFi 时代不同金融服务对比

	传统金融	Fintech	DeFi
货币发行	中央银行	—	POW 或者 POS+
支付&交易	现金	电子现金+中心化网络	数字货币+去中心化网络
借贷	银行	互联网金融平台	数字货币 P2P 借贷平台
资产交易	交易所（如纳斯达克）	传统交易所的线上变化	去中心化链上交易所
投融资	银行、投资机构等	创新型股权、债权平台	金融产品 Token 化（如 IXO）

Defi 的出现和发展为金融提供了新的思路及更多可能，2020 年到 2021 年数字资产市场见证了 DeFi 领域的迅猛发展。DeFi 项目的总体市场市值占比也从 0.9% 上升至了 4.6%。在市值方面，经过这一阶段 DeFi 大爆发，DeFi 市场总市值已经超过 2000 亿美元，在年底相较于之前的最高点增加了 4%。Defi 的爆发充分说明基于区块链公开透明的去中心化金融模型具备巨大的发展潜力，并将在未来行业发展中占据重要角色。DeFi 是人们对于传统金融业求新求变的必然结果，也是新技术对于传统金融业发展的挑战。我们有

理由相信，DeFi 在未来必然会对实体经济产生更多更大的影响。

2.2Web3.0 金融

兼具去中心化和交互性的 Web3.0，得以打造了一个全新的互联网模式。在其中，用户可以绕过中介直接交互。dApp 用户无需许可即可访问金融工具，以点对点的方式交易加密资产，获得参数型保险理赔，通过 NFT 交易可验证所有权的数字艺术品，并且在游戏中赚钱。所有这些活动都可以完全绕过中间方直接展开。

Web3.0 的建设者希望通过这个创新的架构，打造出更加公平和开放的互联网，用户可以在其中直接展开交互和交易。目前，采用了区块链、智能合约和去中心化预言机网络这三种核心技术的 Web3.0 应用已经实现了部分的用例，渗透在房地产、教育、金融、游戏和医疗等各个行业。从这个角度看，Web3.0 代表了一种打破这种垄断控制的愿景——基于区块链技术构建的 Web3 平台和应用程序不会由巨头所有，而是由每个用户拥有，他们将通过帮助开发和维护这些服务来获得所有权。

Web3.0 的愿景是以开源协议为基础，以商业作为接口，提供方便的访问和其他更多特性。Web3.0 是一个对所有用户开放的互联网，建立在开放的协议和透明的区块链网络上。消费者与这些协议交互的方式可能是通过新型应用，这种应用提供了与底层技术交互的便利方法。

Web3.0 将要重新设计现有的互联网服务和产品，使其能够利好于大众而不仅仅是企业巨头。当然，数据仍然会被用来驱动决策，但不会被用来剥削消费者。数据权利将受到保护，而不仅仅是为了追逐利润。激励机制和市场机制将有助于确保信息的可信度和可验

证性。同时，Web3.0 的世界将优先考虑个人的主权，而不是世界上富有的精英和寻租者。

Web3.0 在金融领域的应用将超越我们对目前金融行业的想象。在 Web3.0 的 Defi（去中心化金融）里，不再有一个中心化的金融机构来对你的资质进行审查，无论是你一个被赶出家门的流浪汉，还是一个身穿西装的华尔街大佬，部署在区块链上的智能合约都将无条件地接收你的数字资产并将稳定币借给你。

区块链和智能合约没有感情，它对所有人一视同仁，它只认链上记在的不可篡改的记录以及实实在在的数字资产。在区块链上做生意，也可以几乎“去信任”。这种行为又被 Fred Wilson 称为开放金融（open finance）。

2.3 元宇宙

元宇宙（Metaverse）不是特指某一款应用或产品，它是一个概念，即通过数字化形态承载的平行宇宙。这就意味着我们通过元宇宙可以有第二重、第三重、第 N 重人生，在这里面我们可以有各种全新的身份、资产、社会关系。我们可以在元宇宙中进行完整的生活和社会活动，相当于给我们的生命拓展了长度。

尽管行业内对于元宇宙的最终形态没有细致的描述，但通过细化其特征我们依然能够确定元宇宙的四大核心属性：

（1）同步和拟真。虚拟空间与现实社会保持高度同步和互通，交互效果接近真实。同步和拟真的虚拟世界是原宇宙构成的基础条件，这意味着现实社会中发生的一切事件将同步于虚拟世界，同时用户在虚拟的元宇宙中进行交互时能得到接近真实的 反馈信息。

（2）开源和创造。开源同时意味着技术开源和平台开源，元宇宙通过制定“标准” 和

“协议”将代码进行不同程度的封装和模块化，不同需求的用户都可以在元宇宙进行创造，形成原生虚拟世界，不断扩展元宇宙边际。

（3）永续。元宇宙平台不会“暂停”或“结束”，而是以开源的方式运行并无限期地持续。

（4）闭环经济系统。用户的生产和工作活动将以平台统一的货币被认可，玩家可 以使用货币在平台内消费内容，也可以通过一定比例置换现实货币。经济系统是驱动元宇宙不断前进和发展的引擎。

元宇宙是技术大爆炸时代带来的新世界，而未来十年的现实世界的数字化进程也将大大加快，人类向元宇宙的迁徙速度会大大加快。而且区块链技术打通了虚拟世界和现实的桥梁，它让“元宇宙”从虚拟世界变成了真正的“平行宇宙”。

未来，元宇宙的普及将推动实体经济与数字经济加速深度融合，区块链的技术价值也将在这一进程中逐步显现，它将带来新商业模式，重构分配模式、市场结构、组织形态、产业关系，推动人类走向数字文明新纪元。

2.4 中心化交易所痛点问题

随着数字货币市场的繁荣，为数字货币资产提供流动性和交易撮合需求的数字货币交易所，也随之蓬勃发展。虽然比特币一开始建立是为了去中心化，但在很长一段时间，因为技术实现难、用户使用门槛高，大多数人还是习惯在中心化交易平台交易数字货币。中心化交易所为用户提供账户体系、实名认证、资产充值、资产托管、撮合交易、资产清算、资产兑换等业务服务平台。用户在买卖加密货币时，需要将加密货币或者发币充入交易所，交易所提供流动性，并进行撮合交易、结算等流程。中心化交易所在数千上

万亿市值的数字货币市场中起着举足轻重的作用，选择这个赛道创业者也很多，他们无形中对数字货币的发展起了推动作用。

中心化交易所由企业进行运作，需要管理每个用户注册时的身份信息、账号、交易信息、数字货币余额等。交易需要由中心化的服务器进行撮合成交，并完成记账和对账的工作。整个过程的规则会由官方进行公示，但其中的操作过程都是黑箱，只有交易所内部知情。如果出现了某些 bug 导致交易出错或者黑客攻击导致资金被盗的情况，则需要企业承担责任。由于中心化交易所管理用户的数字货币并且是黑箱操作，对于某些小交易所来说，恶意操纵价格导致客损或者拿着用户的钱跑路了也是历史上发生过的事。

近年来，中心化交易所出现资产安全问题、易受政策风险影响、交易所易操控等各种问题，已经越来越被使用者所诟病：

2.3.1 资产安全问题

在中心化交易所中，交易所就是控制用户资金的中心枢纽，因此黑客攻击目标明确。在过去几年间，不断有中心化交易所被盗，导致用户巨大损失。而黑钱更是中心化交易所一个难以根治的痼疾，中心化交易所采用最严格的 KYC 和花费大量客服成本都无法有效治理黑钱。

2.3.2 政策风险影响

中心化交易所在很多国家和地区都面临着巨大的监管压力，如 OKEX 交易所因徐明星事件宣布暂停用户提币； BITMEX 合规问题面临调查；而最近随着中国大陆最严厉的“924通知”出台，火币、币安等中心化加密货币交易所纷纷宣布退出中国大陆市场。俄乌战

争爆发后，Coinbase 封禁了 25000 个俄罗斯用户的钱包，Dmarket 也封禁了俄罗斯和白俄罗斯的用户账户。以上问题都表明了中心化交易所容易受到政策影响，导致用户资产受到威胁性。

2.3.3 交易所易操控

投资者开展合约交易时，在市场走势产生过大变化或交易所暗箱操作，投资者开展的合约交易非常容易爆仓，导致资产损失。受中心化交易所操控影响，交易走势可能存在深夜、交割前夕出现“一根针”的情况，多空双爆。到了市场行情大幅度震荡的情况下，甚至于无法登陆相关的网站、App 开展操作。导致用户资产严重受损，甚至部分投资者因此轻生。

综上所述，即使中心交易所还是占据主流地位，但在交易中监管缺失，资产安全受到威胁，交易的公平性存在一定的威胁等问题的困扰下，其社会信任机制受到莫大挑战，人们中心化交易所已经逐渐失去信心，去中心化交易市场的崛起已经成为必然。

2.5 解决方案

为了解决这里的上述的这些问题，去中心化交易所应运而生。去中心化交易所与中心化交易所不同，主要体现在“技术”与“治理”两个方面。从技术角度来看，去中心化交易所是通过链上的智能合约来实现交易的。而传统的中心化交易所是在链下进行交易。从治理角度来看，去中心化交易所的治理带有开放和社区驱动的属性。而中心化交易所的治理模式与传统公司相同。去中心化交易所的优点当你使用不同的中心化交易所时，需要重复进行账号注册与 KYC 认证。而使用去中心化交易所则不存在这个问题：用户只需要有一个钱包就能使用不同的去中心化交易所。下图所展示的各个去中心化交易所，

都可通过同一个钱包地址自由交易。它们通过智能合约实现交易，将资产托管、撮合交易等直接放在区块链上进行，而账户密钥仍由用户自己控制，安全隐患大大降低。

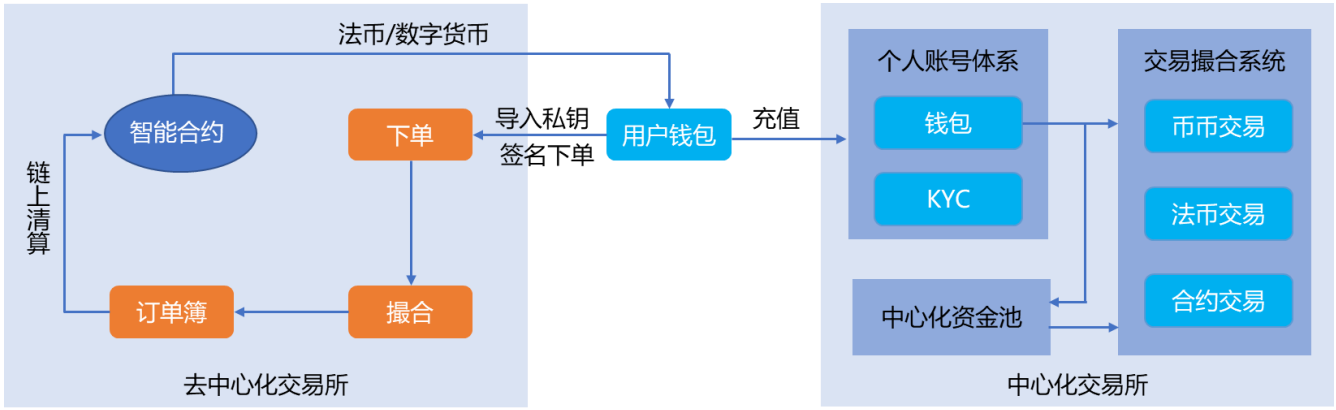

图 2-1：去中心化交易所和传统中心化交易所对比

从上图可见，与中心化交易所不同，去中心化交易所不受单个实体控制，所以在安全性和操控性方面有很大的优势，也不会面临监管政策方面的压力。但是目前很多去中心化交易所虽以去中心化的方式解决了安全问题，在用户体验和交易效率等方面却存在较大的缺陷，比如交易速度慢、黑钱问题仍然难以解决、无法开展法币交易等，仍然无法取代传统的中心化交易所。如果能进一步解决这些问题，去中心化交易在未来必然会取代中心化交易所。

2.6 去中心化交易所市场格局

据 Finbold 发布的一份报告显示，2021 年 1 月 1 日，全球加密货币种类数量为 8153 个，截至 2021 年 12 月 31 日，数量为 16223 个，相比 1 月增加约 98.98%。Finbold 数据显示，2021 年加密行业创造出 8070 种新 Token，平均每天约有 21 种新加密货币在市场上推出。另一项数据显示，2021 年 1-10 月加密市场总计新增约 5000 种加密货币，而 11、12 月有超过 3000 种加密货币进入市场。截至 2021 年 12 月 29 日，全球共有 2.95 亿加密货币用户，相比 2021 年初增长了 178.30%。

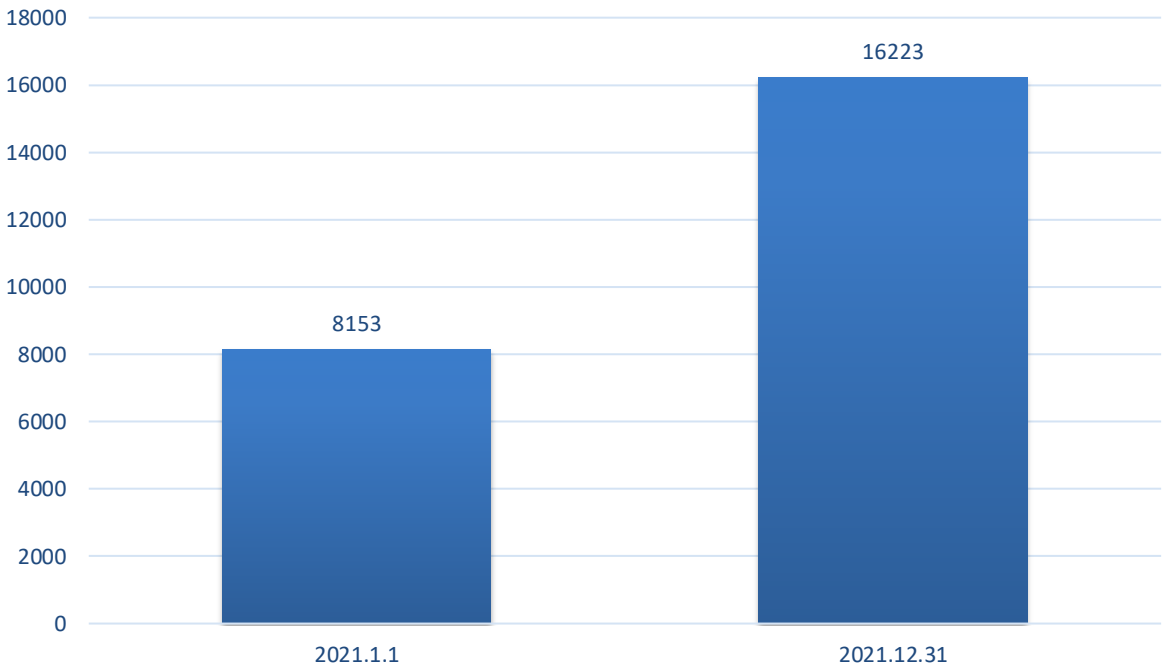

图 2-2：2021 年全球加密数字货币数量变化情况（单位：个）

相对于中心化交易所（CEX）而言，去中心化交易所（DEX）不负责托管用户资产，用户对自己资产有绝对的控制权。去中心化交易所负责提供流动性，撮合交易由智能合约来完成，交易的结算、算清在区块链上完成。

在去中心交易平台领域，Uniswap 依旧保持领先优势，但是新上线的协议抢占了 Kyber 和 dYdX 的市场份额。

Uniswap 的交易量从 2020 年的 578 亿美元（Uniswap 于 2020 年 5 月首次开始促进交易时）增加到 2021 年的 6811 亿美元。第一季度的交易总额为 918 亿美元，而该协议在第四季度结束时的交易量为 2384 亿美元，标志着增加了 160%。

DEX 的市场空间将进一步扩大，二三线 CEX 交易所将受到 DEX 的最直接冲击。随着 DEX 的功能逐步完善，未来将会逐步蚕食传统 CEX 的市场。

3.EOTC 介绍

3.1 EOTC 的定位和服务方向

在这一背景下，新一代去中心化 OTC 交易所 EOTC 诞生。**EOTC 是全球第一个去中心化的 OTC 交易所；全球第一个去中心化的法币交易所；全球第一个去中心化的大宗币币交易所；全球第一个去中心化的大宗法币交易所；元宇宙入口。**

EOTC 由新加坡技术团队和美国技术团队联合开发，依托技术团队在区块链领域的技术积累和安全经验，从多重维度保障全球数字资产用户的资产安全，提供简单便捷、安全可靠的数字资产交易管理服务。与常见的中心化交易所和去中心化币币交易所不同，EOTC 采用的是完全去中心化去信任化的方式，支持各种法币与区块链代币的兑换。我们将为用户实现：

- 公开透明的智能数字货币交易环境和资产管理服务。
- 安全、稳定的资产账户，彻底清理黑钱，还币圈一片净土。
- 主流法币和区块链数字货币的兑换服务。

图 3-1：EOTC 服务方向

EOTC 的愿景是在创建不受任何中心化组织机构控制的高效、公平与无须信任的去中心化 OTC 交易所，让用户在完全自主掌握个人资产的前提下进行高效科学的投资。并通过 EOTC 底层技术实现全球化数字金融互通网络，实现金融自由、网际自由。

未来 EOTC 将不断发展项目生态产品体系，在原有已形成的币币交易、法币交易、杠杆交易的同时，不断扩大生态的覆盖面积，未来业务范围将覆盖理财、ZAPP、万人热聊社交端、支持多币种红包、物联网设备落地、元宇宙等多种产品线于一体的丰富生态，构建强大的 EOTC 生态王国。

3.2 服务内容

3.2.1 交易服务

EOTC 主要为用户提供去中心化的数字货币币币交易、法币交易和杠杆交易。

币币交易：两种不同的加密数字货币之间的交易，以其中一种币作为计价单位去购买其他币种 。用户发起交易请求后，系统按照价格优先时间优先顺序完成撮合交易。EOTC 将开放面向所有主流数字货币币种的配对。

法币交易：包括人民币、美元、日元、欧元、韩元、卢布、加元、英镑等多种法币。用户可在无需充值法币入交易所的情况下进行法币交易，达到完全的去中心化。

杠杆交易：用户抵押数字资产作为保证金后，平台将给予用户自有资金一定倍数的杠杆资金，用户可以以杠杆资金对某一数字货币做多或做空获取利润。

后期 EOTC 将陆续开发各种优质的币种配对，以提高交易所的实用性与流动性。交易所的交易流程如下图所示：

图 3-2：EOTC 交易流程

如果用户发生交易纠纷，将进入仲裁流程：

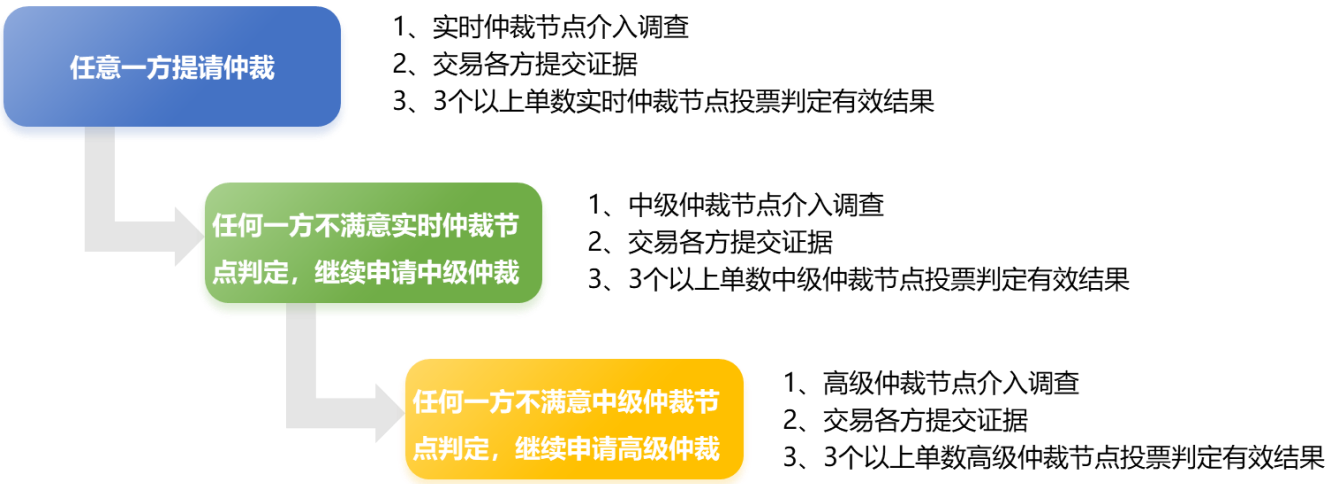

图 3-3：仲裁流程

从上图可以看出，EOTC 将资产的控制权全部交给用户，交易所本身不储存任何资产，也不会对交易过程做任何操控（实际上也无法进行任何操控），交易的订单会在链上做清算，交易纠纷由仲裁节点按照规则执行仲裁。所以整个过程都可以做到公开透明，用户可以完全放心。

为了更好地提供服务，EOTC 建立了承兑商和仲裁节点的角色。

- 承兑商：主要负责用户出入金，承兑商要向交易所质押代币，根据质押数量，并结合信用积分划分等级，不同等级的承兑商有不同的权限。前期将引入几大交易所作为承兑商。

- 仲裁节点：仲裁节点主要是为有交易纠纷的用户起仲裁作用，分实时节点，中级节点，高级节点。仲裁节点按照社区投票制定的仲裁规则执行仲裁，节点信用积分与奖励挂勾。

3.2.2 去中心化金融服务

在交易服务的基础上，EOTC 未来还将提供多样化的去中心金融服务，具体包括：

1、流动性挖矿

在现有的金融系统中，金融服务主要由中央系统控制和调节，无论是最基本的存取转账、还是贷款或衍生品交易。EOTC 的流动性挖矿则希望通过分布式开源协议建立一套具有透明度、可访问性和包容性的点对点金融系统，将信任风险最小化，让参与者更轻松便捷地获得融资，让投资者可以获取稳定的收益。

在 EOTC 流动性挖矿服务中，存款用户存入数字货币，为资金池提供流动性。用户可以以不同的方式与资金进行交互，包括存款、赎回、借款、还款、清算和闪电贷等。借款用户可以根据准备金中可用的存款货币借入资金，并锁定更大的价值作为抵押。借款用户还款后，利息收入在扣除一定额度的手续费后将回报给存款用户。

这个流动池是运行在智能合约上的，没有任何庄家和大户，智能合约保证了账目的真实性，这样不需要记账、审核、防伪、安保等一系列成本，授权、审计、查账等功能也是智能合约自带的，所以可以节约下非常高的运营成本，这些都可以作为补贴给到存款用户手上。

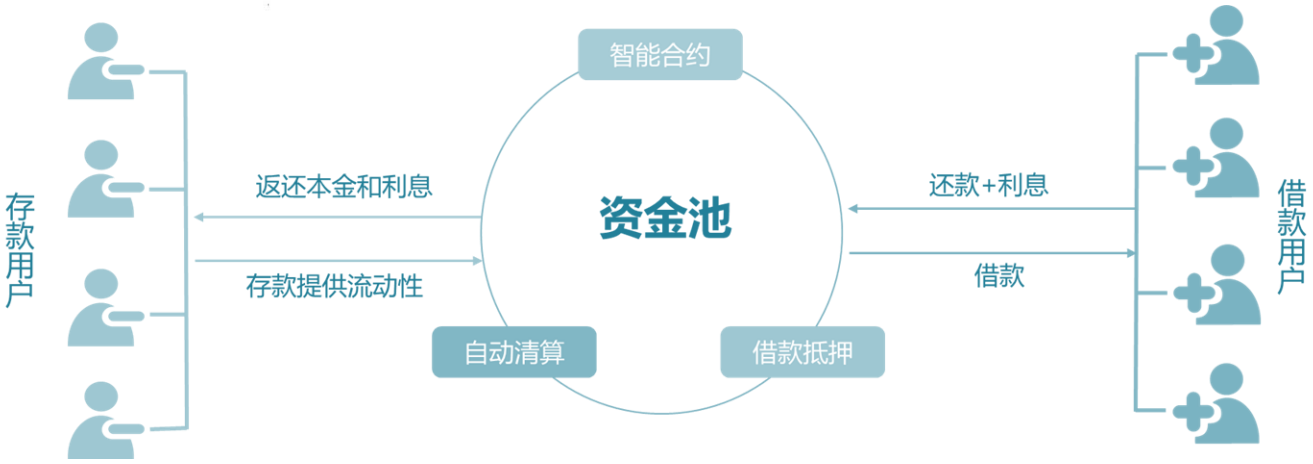

图 3-4：流动性挖矿

EOTC 将以发行的数字货币挖矿搭建完全去中心化元宇宙生态，用户可以在 EOTC 搭建的元宇宙中创建属于自己的 NFT，通过抵押 NFT 获取各类数字货币，进行 Defi 挖矿。

2、跨链闪兑

EOTC 通过 DAO 和底层跨链协议的支持，可以实现 EOTC 发行的数字货币和其他主流数字货币的跨链闪兑，以后还会对接增加各个国家的官方数字货币，这些金融服务已经接近于银行的功能，无论是区块链项目方还是其用户，都可以在 EOTC 上进行快捷方便的数字资产流通。货币的核心功能就是流通，EOTC 的跨链闪兑流通功能让大量的数字资产能够广泛地流通起来，数字资产的价值也就越来越显著，每个普通人都能感受到区块链技术给我们带来的便利。

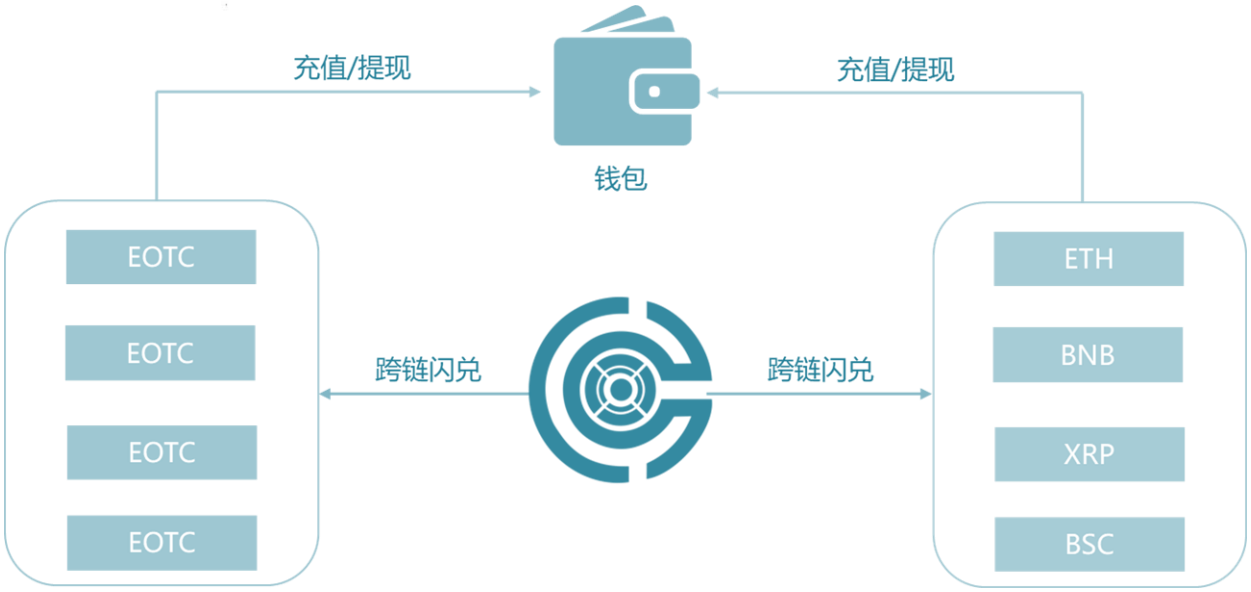

图 3-5：跨链闪兑

3.2.3 社交服务

EOTC 的社交服务包括 SocialFi 和隐秘通信等多种服务。

1、SocialFi

SocialFi 是 EOTC 社交服务的核心服务内容。在 EOTC 平台上将会有各种各样的社交应用，如交友、聊天、直播、社群、游戏等。与传统社交最大的区别在于 EOTC 平台上的游戏都融入了 SocialFi，真正实现传统社交无法实现的边玩边赚。

MetaSocial 平台的 SocialFi 解决了以下问题：

- 数据的归属权问题。也就是每个人生产的内容是绝对属于自己的，而非传统社交平台一样储存在运营商服务器中。EOTC 上的每一个虚拟资产都可以转换为 NFT，这是区块链上独特的、稀有的、不可替代的数字资产，允许用户轻松验证真实性和所有权。
- 利益分配的问题。对于目前许多 UGC 而言，创作者都是为爱发电，很难维持持续创作的动力，在 EOTC 平台上，每个人都是自己内容的直接受益者，用户所有创造带来的价值，将完完全全归属于他们自己所有，并可以体现为平台代币——EOTC，进行

高效地流通。

- 隐私问题。中心化服务器管理下的审查问题，使每个人的隐私都难以得到保障，而 EOTC 的数据存储于分布式存储系统上，拥有无法破解的高强度加密技术，可以最大程度的避免这个问题。

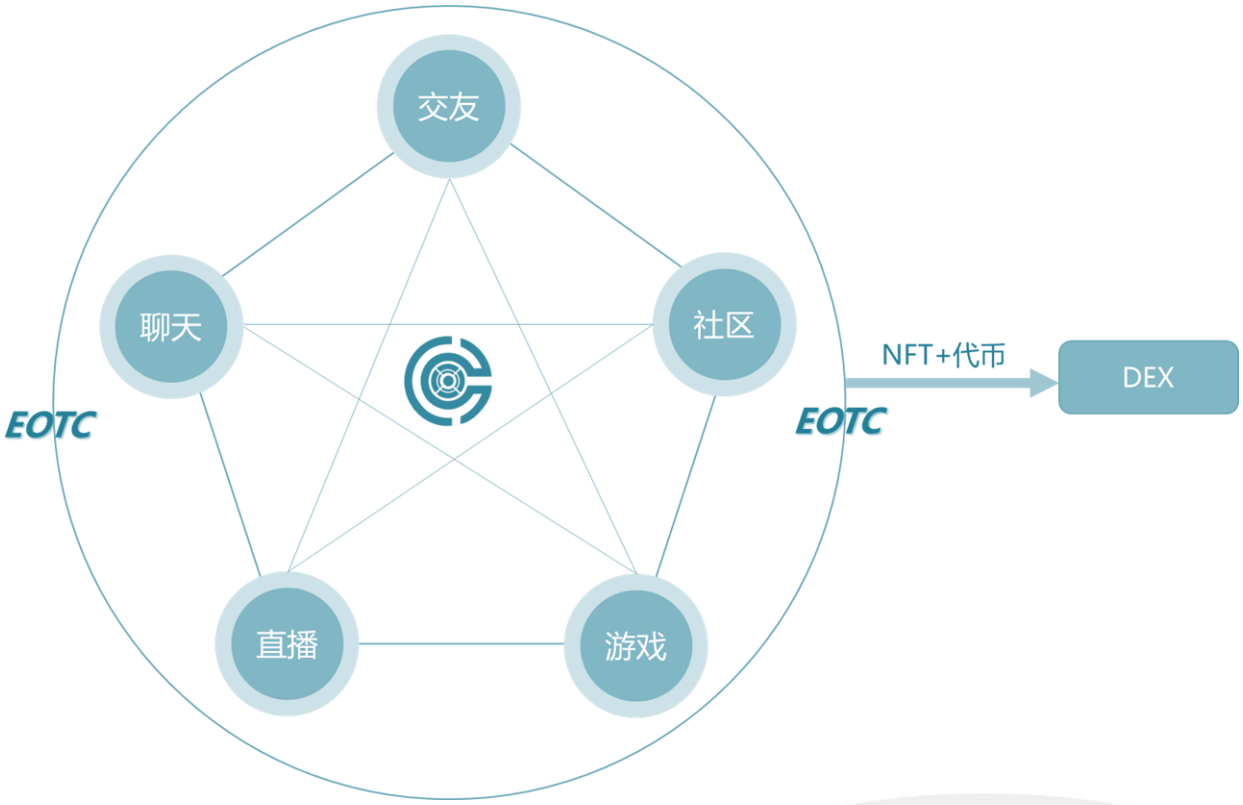

图 3-6：SocialFi

2、隐秘通信

EOTC 通过应用区块链技术的底层加密算法为通讯讯息进行加密，以从未被破解的区块链加密算法为支撑，确保信息无法破译，只有通信双方能够解密通讯内容。加密用的私钥由用户自己使用的客户端生成，除了用户自己其他人无法获知。

通过点对点的信息传送，确保信号不被中间服务商（即使 EOTC 也不能）截留破解。由私钥生成的通讯地址，在客户端显示的只是一串唯一的字符代码，用于识别不同用户的通讯地址，而不记载任何的用户信息，因此不会泄露或绑定联系人身份。

用户的历史聊天记录经过私钥加密后存储至 fil 分布式存储。如果用户私钥被窃取，除非黑客知道聊天记录的保存地址，或可以复制调用聊天记录，否则不会泄露用户任何历史信息。

前台部分，用户可以自由传输文字、图片、短视频、语音、位置、红包等内容，使用起来和传统社交软件一样方便，而且更加适合应用在对隐私要求比较高的领域。

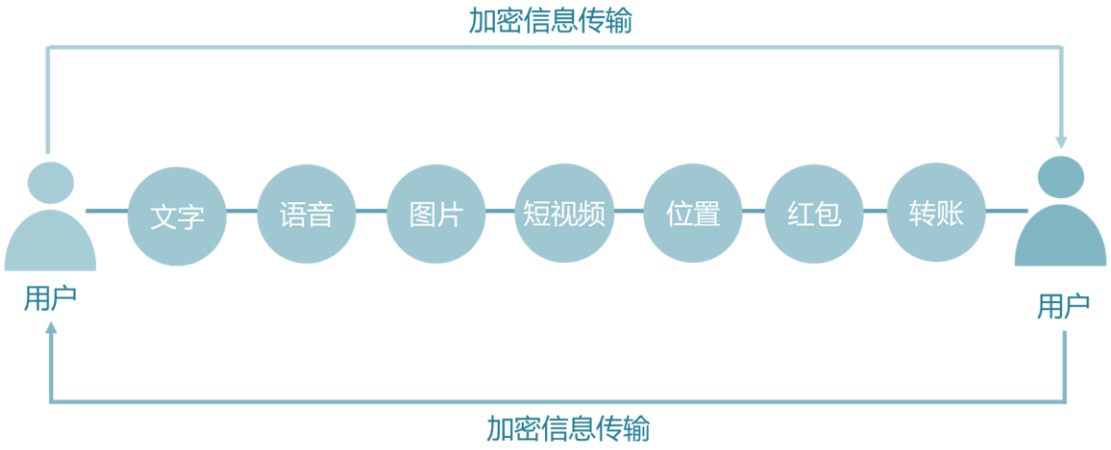

图 3-7：隐秘通信

3.2.4 元宇宙

EOTC 未来将成为一个去中心化的元宇宙，在 EOTC 元宇宙中将会承载一个虚拟的世界，为用户提供多样化的服务，我们初步规划的服务包括链游孵化、NFT 服务平台、用户社区等。

1、链游孵化和游戏分发服务

EOTC 将构建一个元宇宙链游的孵化平台，平台将为去中心化的链游开发者提供开发技术支持、宣传推广、金融流动支持等多方面的服务，如工具套件 SDK 以及游戏代币的经

济建议等，让开发者可以一站式获取各类资源，使得传统的游戏开发商可以快速及时的切入 Gamefi 游戏开发市场，顺利开发出满足市场需求的链游，同时丰富了 EOTC 平台的生态内容。在 EOTC 平台上开发的链游，EOTC 都将会为其提供分发服务，EOTC 通过去中心化社区让登录平台的游戏都能够享受公平公正的待遇，保障每一个游戏的排名和曝光。EOTC 还通过通证经济让开发者获得除了游戏销售之外的其他盈利渠道，增加开发者的收益，开发者可以使用法币或者 EOTC 的平台币支付各种费用，也可以通过平台合作的交易所进行兑换，方便快捷。

尤为重要的是，EOTC 平台上的游戏都融入了 Gamefi，真正实现传统游戏无法实现的即玩即赚。游戏玩家可以通过打怪、抽取盲盒、做任务、比赛等方式获得游戏装备、经验、NFT 卡片、通证与奖励，而部分获得的道具可以实时兑换成可交易的资产。在 EOTC 中所进行游戏、充值、氪金等任何行为所带来的价值，将完完全全归属于玩家本身所有，并可以体现为代币，进行高效地流通。

用户可以在 EOTC 平台上尽情玩乐，放松和赚钱。平台上的各类游戏旨在针对不同年龄、性别、职业和社会阶层的不同玩家提供吸引力，我们当然希望为尽可能多的加密货币爱好者带来不同的服务，但我们也希望吸引非加密货币爱好者以及游戏玩家进入到 EOTC 的游戏之中。在游戏的世界里玩家不仅可以获得优秀的游戏体验，还能获取收益。

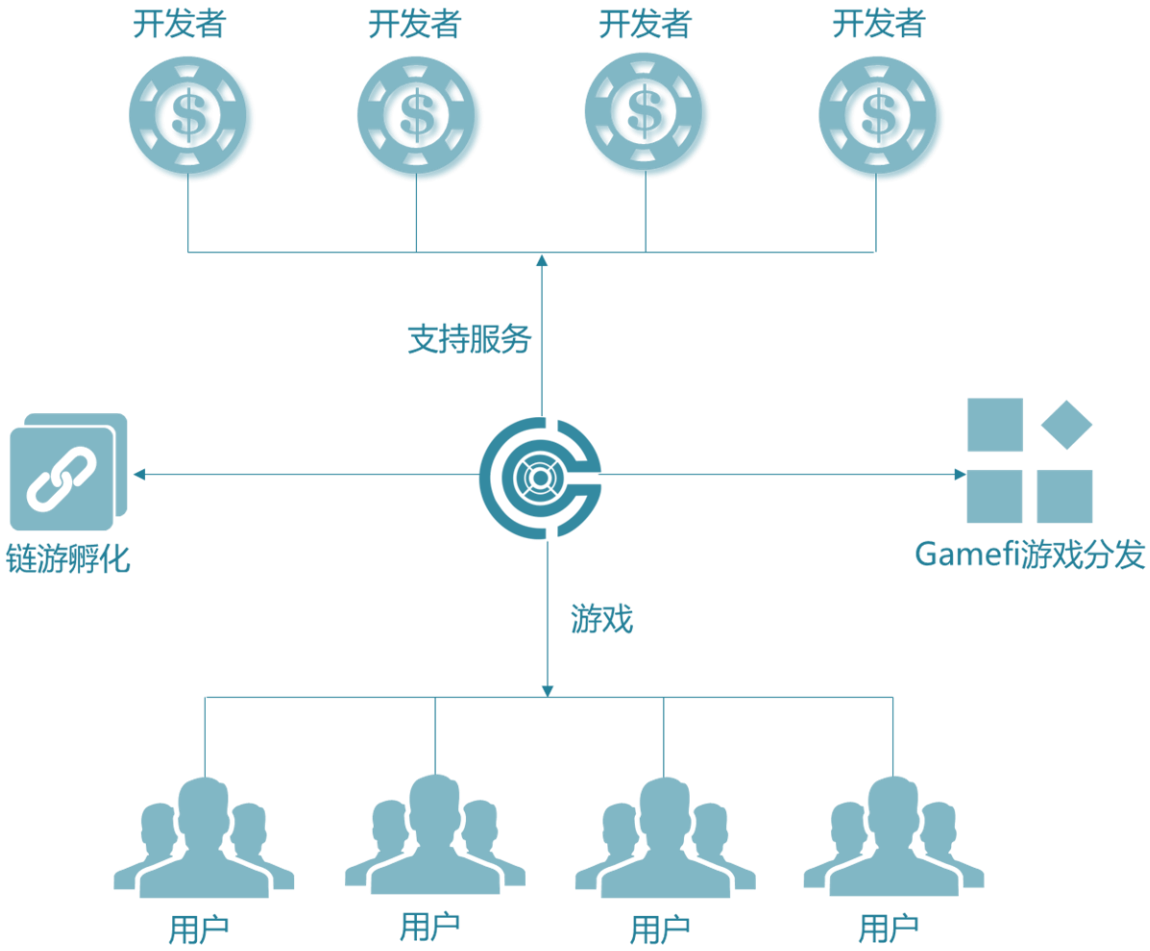

图 3-8：链游孵化和游戏分发服务

2、NFT 服务平台

EOTC 将建设 NFT 服务平台，建立完整的交易与资产合成服务系统，逐步打造完善的 NFT 生态体系。

NFT 服务平台的主要服务内容包括 NFT 的铸造、交易、金融、展会等多样化的服务。

NFT 铸造：有别于大多数 NFT“发行方制作，用户购买”这种自上而下的 ICO 逻辑，EOTC 中的 NFT 获取方式类似于 LOOT，是一种更加去中心化的 NFT 铸造方式，任何人都有机会参与铸造发行，用户只要支付手续费，即可生成各种 NFT。为了产生随机的稀缺性，合约会为所有者分配一个 ID。任何人都可以生成 NFT，然后与其他协议进行组合

连接、扩展，类似于 DeFi 堆积木，可以在 MVP（最小化可行产品）的基础上不断创造和建设。

交易：平台将通过合作机构审计的 NFT 产品信息公开展示给客户，客户可以通过拍卖或者抽取盲盒的方式进行交易。

金融：平台会提供一系列和 NFT 相关的金融服务，比如交易险，租借和贷款等。最大限度降低 NFT 的托管收藏与交易过程带来的不可预见风险并提供用户的体验性。

展会：发起线下或线上展会，NFT 所有者获得相应收益分配。

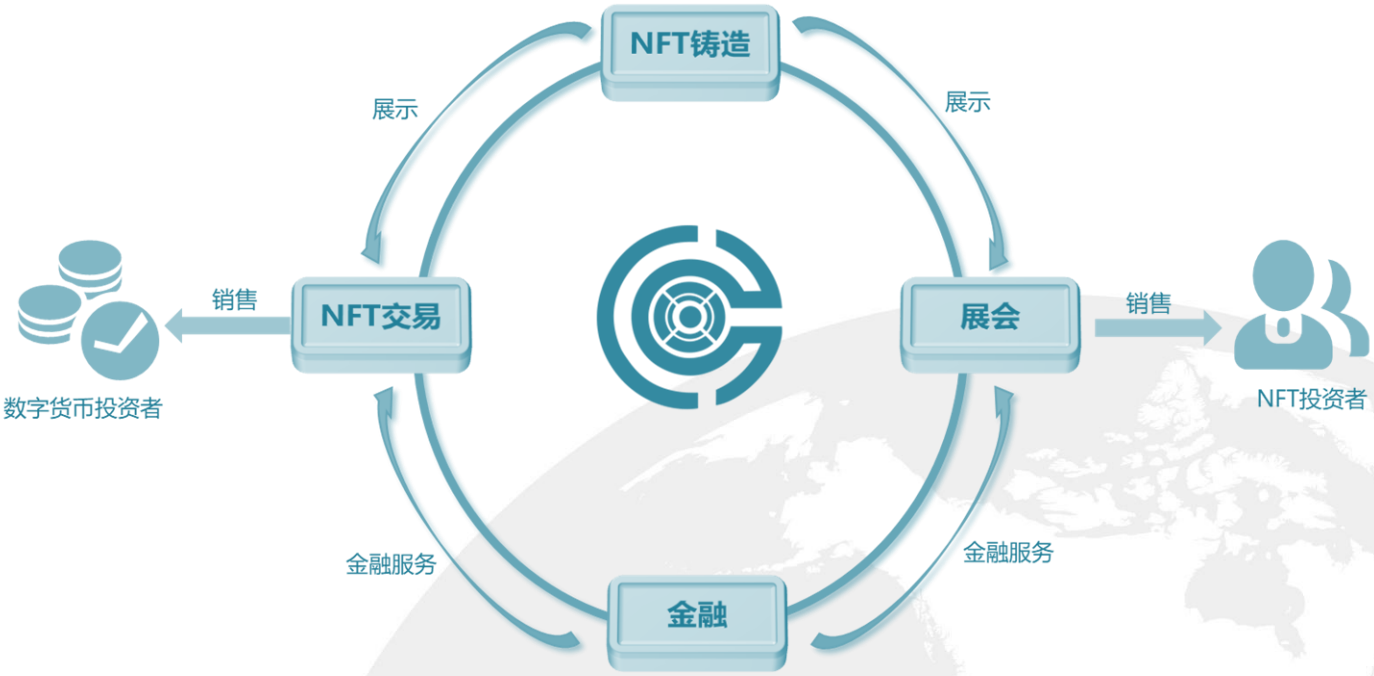

图 3-9：NFT 服务平台

2、用户社区

EOTC 元宇宙中会让用户在区块链创造的世界中自由地开展娱乐和社交活动。在 EOTC 开展的游戏或者社交娱乐中，玩家需要不断的投入心血和资源，来获得更多种类的有价值

的虚拟物品，整个过程充满了探索和收集的趣味性。玩家的真实情感投射到 EOTC 元宇宙中，以社交娱乐的方式来实现他们在现实生活中无法实现的梦想。EOTC 以娱乐的方式来创造世界，改变世界，并推动区块链这项伟大的技术的进步。在 EOTC 元宇宙中，每个用户都可以拥有自己的领土，这些领土不只属于项目方，更是属于各位持有者的一片天堂，用户在这里可以自由讨论各类问题，互相沟通，加深了解。同时，社区也可以让对数字资产兴趣的群体开展资产走向、投资趋势等各方面的专题讨论，让整个平台的参与人群越来越多，形成良好的氛围。

图 3-10：用户社区

3.3 EOTC 特点

EOTC 有着以下的特点：

3.3.1 法币交易

目前市场上的去中心化交易所基本都不支持法币交易，因为法币交易存在着黑钱问题，这一问题在经过严格身份认证的中心化交易所都难以根治解决，而去中心化交易所将用户的资产所有权完全交给用户，就更加难以解决这个问题。EOTC 针对这个问题，首创了行业反黑钱解决方案（因为涉及到机密，本白皮书中对该方案的技术细节不予公开）。通过这一方案，可以彻底杜绝黑钱的困扰。在反黑钱方案的支持下，EOTC 交易所成为全球

首个支持法币交易的去中心化交易所。在中心化交易所受到政策监管压力的大背景下，EOTC 有望成为全球法币交易用户的避风港。

3.3.2 完善的制度设计

去中心化交易中往往存在用户体验不佳的问题，究其原因在于交易所没有管控手段来规避用户的违规行为，针对这个问题，EOTC 设计了一整套制度，不仅包括仲裁节点，还引入了信用积分机制，通过智能合约为用户、承兑商和仲裁节点的建立了信用积分账户，用户的信用积分将决定他的交易等级，有足够信誉积累的用户将可以获得更多的优惠和便利。而信用积分降低到一定程度将失去在交易所中的角色权利。

3.3.3 高安全性

EOTC 采用美国硅谷领先全球的区块链 4.0 研发技术及顶尖“黑匣子”级别防护架构，抵御黑客攻击；用户自持私钥，用户的托管资产可以自由转移无需任何人审批，也不用担心黑客盗取、丢币等问题发生，安全上具有足够的保障，极大地降低安全隐患。

3.3.4 高性能支持

EOTC 的交易系统目前部署在以太坊公链上，采用多边撮合技术，并经权威测评中心认证。EOTC 后续将陆续部署在各大公链上，通过跨链技术和瞬时交易技术，EOTC 的订单处理速度将达到 100 万单/秒。

3.3.5 全平台支持

EOTC 将提供对全平台客户端的支持，具体包括：Windows、Android、IOS、H5、WEB 等。

3.4 发展规划

EOTC 未来将成为一个去中心化金融服务的元宇宙，但我们也并非一蹴而就。项目团队将脚踏实地，根据技术的发展成熟程度对项目进行不断迭代升级，根据我们的规划，未来 3 年项目的发展将经历多个版本，具体如下：

- EOTC1.0：2021-2022

实现交易所基本功能，开放常见币种的去中心化交易对，包括币币交易和法币交易，如 CNY/USDT、CNY/BTC、CNY/ETH、USDT/BTC、USDT/ETH 等交易对开放；

- EOTC2.0：2022-2023

开发其他法币和其他链上合约、开发以太坊二层等高速低费交易所、合约交易所等内容，推出期货合约市场，支持包括 TRC-20、 ERC-20、跨链资产及各类链下资产交易；

- EOTC3.0：2023-2024

开发跨链桥元宇宙入口、铸造链上稳定币；基于 LP 资产的去中心化贷款和再投资业务上线，打造多元化业务生态；

- EOTC4.0：2024-2025

全面布局元宇宙生态，建设社交、娱乐、物联网、去中心化金融等多类型服务，实现去中心化金融服务元宇宙目标。

4.技术阐述

4.1 系统逻辑架构

图 4-1 所示为 EOTC 红牛交易所设计的系统逻辑架构图。

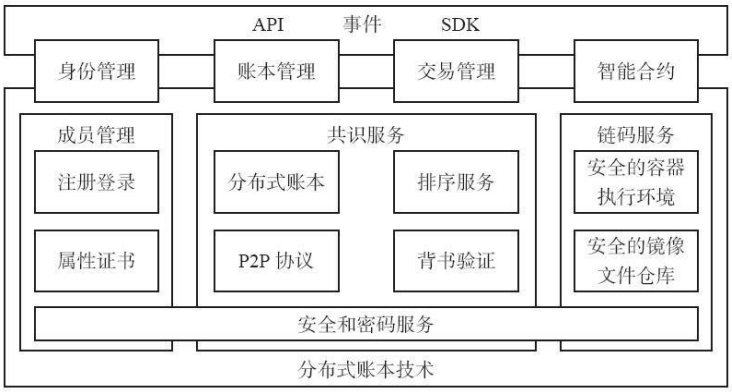

图 4-1 系统逻辑架构图

图 4-1 所示的系统逻辑架构图是从不同角度来划分的，上层从应用程序的角度，提供了标准的 gRPC 接口，在 API 的基础之上封装了不同语言的 SDK，包括 Golang、Node.js、Java、Python 等，开发人员可以利用 SDK 开发基于区块链的应用。区块链强一致性要求，各个节点之间达成共识需要较长的执行时间，也是采用异步通信的模式进行开发的，事件模块可以在触发区块事件或者链码事件的时候执行预先定义的回调函数。下面分别从应用程序和底层的角度分析应该关注的几个要素。

4.1.1 应用程序角度

1、身份管理

用户注册和登录系统后，获取到用户注册证书（ECert），其他所有的操作都需要与用户证书关联的私钥进行签名，消息接收方首先会进行签名验证，才进行后续的消息处理。网络节点同样会用到颁发的证书，比如系统启动和网络节点管理等都会对用户身份进行认证和授权。

2、账本管理

授权的用户是可以查询账本数据（ledger）的，这可以通过多种方式查询，包括根据区块号查询区块、根据区块哈希查询区块、根据交易号查询区块、根据交易号查询交易，还可以根据通道名称获取查询到的区块链信息。

3、交易管理

账本数据只能通过交易执行才能更新，应用程序通过交易管理提交交易提案（Proposal）并获取到交易背书（Endorsement）以后，再给排序服务节点提交交易，然后打包生成区块。SDK 提供接口，利用用户证书本地生成交易号，背书节点和记账节点都会校验是否存在重复交易。

4、智能合约

实现“可编程的账本”（Programmable Ledger），通过链码执行提交的交易，实现基于区块链的智能合约业务逻辑。只有智能合约才能更新账本数据，其他模块是不能直接修改状态数据（World State）的。

4.1.2 底层角度

下面的内容是从 EOTC 底层的角度来看，如何实现分布式账本技术，给应用程序提供区块链服务。

1、成员管理

MSP（Membership Service Provider）对成员管理进行了抽象，利用 KPI(Public Key

Infrastructure）对成员身份进行认证，验证成员用户提交请求的签名。结合 Fabric-CA 或者第三方 CA 系统，提供成员注册功能，并对成员身份证书进行管理，例如证书新增和撤销。注册的证书分为注册证书（ECert）、交易证书（TCert）和 TLS 证书（TLS Cert），它们分别用于用户身份、交易签名和 TLS 传输。

2、共识服务

在分布式节点环境下，要实现同一个链上不同节点区块的一致性，同时要确保区块里的交易有效和有序。共识机制由 3 个阶段完成：客户端向背书节点提交提案进行签名背书，客户端将背书后的交易提交给排序服务节点进行交易排序，生成区块和排序服务，之后广播给记账节点验证交易后写入本地账本。网络节点的 P2P 协议采用的是基于 Gossip 的数据分发，以同一组织为传播范围来同步数据，提升网络传输的效率。

3、链码服务

智能合约的实现依赖于安全的执行环境，确保安全的执行过程和用户数据的隔离。EOTC 采用 Docker 管理普通的链码，提供安全的沙箱环境和镜像文件仓库。其好处是容易支持多种语言的链码，扩展性很好。Docker 的方案也有自身的问题，比如对环境要求较高，占用资源较多，性能不高等，实现过程中也存在与 Kubernetes、Rancher 等平台的兼容性问题。

4、安全和密码服务

安全问题是企业级区块链关心的问题，尤其在关注交易安全的项目中。其中底层的密码学支持尤其重要，EOTC 专门定义了一个 BCCSP（BlockChain Cryptographic Service Provider），使其实现密钥生成、哈希运算、签名验签、加密解密等基础功能。BCCSP

是一个抽象的接口，默认是软实现的国际算法，目前社区和较多的厂家都在实现国际加密的算法和 HSM（Hardware Security Module）。

4.2 网络节点架构

节点是区块链的通信主体，是一个逻辑概念。多个不同类型的节点可以运行在同一物理服务器上。有多种类型的节点：客户端、Peer 节点、排序服务节点和 CA 节点。图 4-2 所示为网络节点架构图。接下来详细地解释图 4-2 所示的不同节点的类型。

4.2.1 客户端节点

客户端或者应用程序代表由 终用户操作的实体，它必须连接到某一个 Peer 节点或者排序服务节点上与区块链网络进行通信。客户端向背书节点（Endorser）提交交易提案（Transaction Proposal），当收集到足够背书后，向排序服务广播交易，进行排序，生成区块。

4.2.2Peer 节点

所有的 Peer 节点都是记账节点（Committer），负责验证从排序服务节点区块里的交易，维护状态数据和账本的副本。部分节点会执行交易并对结果进行签名背书，充当背书节点的角色。背书节点是动态的角色，是与具体链码绑定的。每个链码在实例化的时候都会设置背书策略，指定哪些节点对交易背书后才是有效的。也只有在应用程序向它发起交易背书请求的时候才是背书节点，其他时候就是普通的记账节点，只负责验证交易并记账。

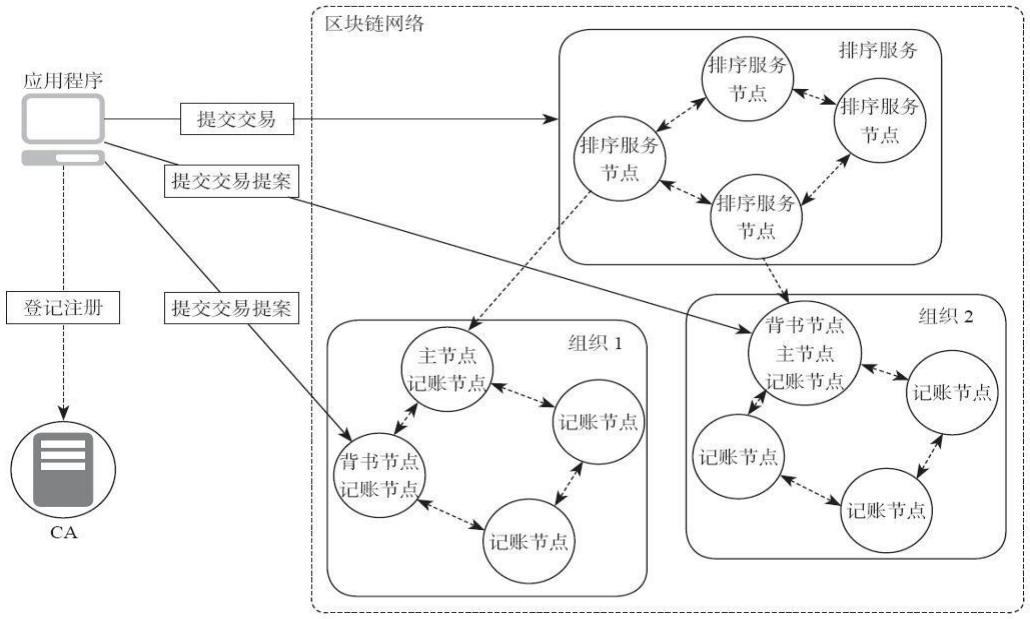

图 4-2 网络节点架构图

图 4-2 所示的 Peer 节点还有一种角色是主节点（Leader Peer），代表的是和排序服务节点通信的节点，负责从排序服务节点处获取 新的区块并在组织内部同步。可以强制设置为主节点，也可以动态选举产生。在图 4-2 中还可以看到，有的节点同时是背书节点和记账节点，也可以同时是背书节点、主节点和记账节点，也可以只是记账节点。

3.2.3 排序服务节点

排序服务节点（Ordering Service Node 或者 Orderer）接收包含背书签名的交易，对未打包的交易进行排序生成区块，广播给 Peer 节点。排序服务提供的是原子广播（Atomic Broadcast），保证同一个链上的节点接收到相同的消息，并且有相同的逻辑顺序。

排序服务的多通道（MultiChannel）实现了多链的数据隔离，保证只有同一个链的 Peer 节点才能访问链上的数据，保护用户数据的隐私。

排序服务可以采用集中式服务，也可以采用分布式协议。可以实现不同级别的容错处理，目前正式发布的版本只支持 Apache Kafka 集群，提供交易排序的功能，只实现 CFT（Crash Fault Tolerence，崩溃故障容错），不支持 BFT（Byzantine Fault Tolerance，拜占庭容错）。

4.2.4CA 节点

CA 节点是 EOTC 的证书颁发机构（Certificate Authority），由服务器和客户端组件组成。CA 节点接收客户端的注册申请，返回注册密码用于用户登录，以便获取身份证书。在区块链网络上所有的操作都会验证用户的身份。CA 节点是可选的，可以用其他成熟的第三方 CA 颁发证书。

4.3 典型交易流程

图 4-3 所示为 EOTC 典型的交易流程图。

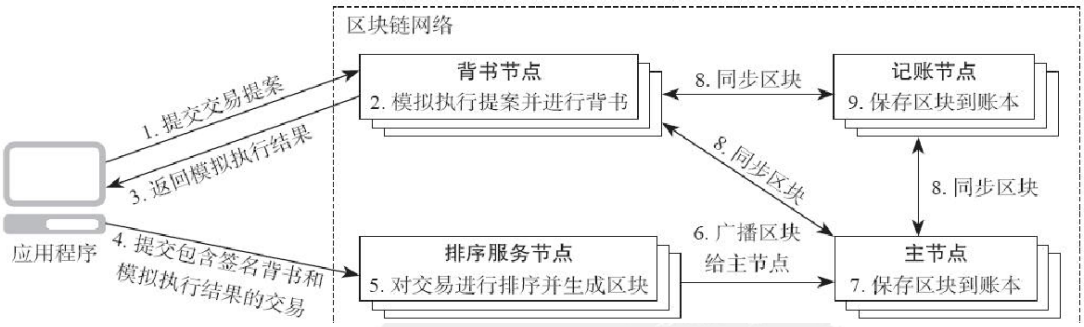

图 4-3 交易流程总图

从上一节的网络节点架构中，我们已经了解到基于 EOTC 的区块链应用中涉及几个节点角色：应用程序、背书节点、排序服务节点和主节点。在图 4-3 中，假定各节点已经提前颁发好证书，且已正常启动，并加入已经创建好的通道。后面的步骤介绍在已经实例化了的链码通道上从发起一个调用交易到 终记账的全过程。

4.3.1 创建交易提案并发送给背书节点

使用应用程序构造交易提案，SignedProposal 的结构如下所示：

```
SignedProposal：{
    ProposalBytes（Proposal）：{
        Header：{
            ChannelHeader：{
Type："HeaderType_ENDORSER_TRANSACTION",
                TxId：TxId,
                Timestamp：Timestamp,
                ChannelId：ChannelId,
                Extension（ChaincodeHeaderExtension）：{
                    PayloadVisibility：
PayloadVisibility,
                    ChaincodeId：{
                        Path：Path,
                        Name：Name,
                        Version：Version
                    }
                },
                Epoch：Epoch
            },
            SignatureHeader：{
                Creator：Creator,
                Nonce：Nonce
            }
        },
        Payload：{
            ChaincodeProposalPayload：{
                Input（ChaincodeInvocationSpec）：{
```

```
                ChaincodeSpec：{
                    Type：Type,
                    ChaincodeId：{
                        Name：Name
                    },
                    Input (ChaincodeInput):{
                        Args：[]
                    }
                }
            },
            TransientMap：TransientMap
        }
    }
},
Signature：Signature
}
```

我们来看看上面的结构，SignedProposal 是封装了 Proposal 的结构，添加了调用者的签名信息。背书节点会根据签名信息验证其是否是一个有效的消息。

Proposal 由两个部分组成：消息头和消息结构。

消息头（Header）也包含两项内容。

1、通道头（ChannelHeader）：通道头包含了与通道和链码调用相关的信息，比如在哪个通道上调用哪个版本的链码。TxId 是应用程序本地生成的交易号，跟调用者的身份证书相关，可以避免交易号的冲突，背书节点和记账节点都会校验是否存在重复交易。

2、签名头（SignatureHeader）：签名头包含了调用者的身份证书和一个随机数，用于消息的有效性校验。

应用程序构造好交易提案请求后，选择背书节点执行并进行背书签名。背书节点是链码背书策略里指定的节点。有一些背书节点是离线的，其他的背书节点可以拒绝对交易进行背书，也可以不背书。应用程序可以尝试使用其他可用的背书节点来满足策略。应用程序以何种顺序给背书节点发送背书请求是没有关系的，正常情况下背书节点执行后的结果是一致的，只有背书节点对结果的签名不一样。

4.3.2 背书节点模拟交易并生成背书签名

背书节点在收到交易提案后会进行一些验证，包括：

- 交易提案的格式是否正确；
- 交易是否提交过（重复攻击保护）；
- 交易签名有效（通过 MSP）；
- 交易提案的提交者在当前通道上是否已授权有写权限。

验证通过后，背书节点会根据当前账本数据模拟执行链码中的业务逻辑并生成读写集（RwSet），其中包含响应值、读写集等。在模拟执行时账本数据不会更新。而后背书节点对这些读写集进行签名成为提案响应（Proposal Response），然后返回给应用程序。ProposalResponse 的结构如下：

```
ProposalResponse：{
    Version：Version,
    Timestamp：Timestamp,
    Response：{
        Status：Status,
  Message：Message,
        Payload：Payload
    },
```

```
Payload（ProposalResponsePayload）：{
    ProposalHash：ProposalHash,
    Extension（ChaincodeAction）：{
        Results（TxRwSet）：{
            NsRwSets（NsRwSet）：[
                NameSpace：NameSpace,
                KvRwSet：{
                    Reads（KVRead）：[
                        Key：Key,
                        Version：{
                            BlockNum：BlockNum,
                            TxNum：TxNum
                        }
                    ],
RangeQueriesInfo（RangeQueryInfo）：[
                    StartKey：StartKey,
                        EndKey：EndKey,
                        ItrExhausted：ItrExhausted,
                        ReadsInfo：ReadsInfo
                    ],
                    Writes（KVWrite）：[
                        Key：Key,
                        IsDelete：IsDelete,
                        Value：Value
                    ]
                }
]
        Events（ChaincodeEvent）：{
            ChaincodeId：ChaincodeId,
            TxId：TxId,
            EventName：EventName,
```

```
            Payload：Payload
        }
        Response：{
            Status：Status,
            Message：Message,
            Payload：Payload
        },
        ChaincodeId：ChaincodeId
    }
},
Endorsement：{
    Endorser：Endorser,
    Signature：Signature
}
}
```

返回的 ProposalResponse 中包含了读写集、背书节点签名以及通道名称等信息。

4.3.3 收集交易的背书

应用程序收到 ProposalResponse 后会对背书节点签名进行验证，所有节点接收到任何消息后都是需要先验证消息合法性的。如果链码只进行账本查询，应用程序会检查查询响应，但不会将交易提交给排序服务节点。如果链码对账本进行 Invoke 操作，则须提交交易给排序服务进行账本更新，应用程序会在提交交易前判断背书策略是否满足。如果应用程序没有收集到足够的背书就提交交易了，记账节点在提交验证阶段会发现交易不能满足背书策略，标记为无效交易。

如何选择背书节点呢？目前 fabric-sdk-go 默认的实现是把配置文件选项 channels.mychannel.peers（其中的 mychannel 需要替换成实际的通道名称）里的节

点全部添加为背书节点，需要等待所有背书节点的背书签名。应用程序等待每个背书节点执行的超时时间是通过配置文件选项 client.peer.timeout.connection 设置的，配置文件的示例给出的是 3 秒，根据实际情况调整，如果没有设置就是 5 秒的默认值。

4.3.4 构造交易请求并发送给排序服务节点

应用程序接收到所有的背书节点签名后，根据背书签名调用 SDK 生成交易，广播给排序服务节点。生成交易的过程比较简单，确认所有的背书节点的执行结果完全一致，再将交易提案、提案响应和背书签名打包生成交易。交易的结构如下：

```
Envelope：{
    Payload：{
        Header：{
            ChannelHeader：{
Type："HeaderType_ENDORSER_TRANSACTION",
                TxId：TxId,
                Timestamp：Timestamp,
                ChannelId：ChannelId,
                Extension（ChaincodeHeaderExtension）：{
                    PayloadVisibility：
PayloadVisibility,
                    ChaincodeId：{
                        Path：Path,
                        Name：Name,
                        Version：Version
                    }
                },
                Epoch：Epoch
            },
            SignatureHeader：{
```

```
                Creator：Creator,
                Nonce：Nonce
            }
        },
        Data（Transaction）：{
            TransactionAction：[
                Header（SignatureHeader）：{
                    Creator：Creator,
                    Nonce：Nonce
                },
                Payload（ChaincodeActionPayload）：{
                    ChaincodeProposalPayload：{
Input（ChaincodeInvocationSpec）：{
ChaincodeSpec：{
                                Type：Type,
                                ChaincodeId：{
                                    Name：Name
                                },
                                Input（ChaincodeInput）：
                                {
                                    Args：[]
                                }
                            }
                        },
                        TransientMap：nil
                    },
                    Action（ChaincodeEndorsedAction）：{
Payload（ProposalResponsePayload）：{
                            ProposalHash：ProposalHash,
Extension（ChaincodeAction）：{
                                Results（TxRwSet）：{
```

NsRwSets（NsRwSet）：[

NameSpace：NameSpace,

KvRwSet：{

Reads（KVRead）：[

Key：Key,

Version：{

BlockNum：BlockNum,

TxNum：TxNum

}

],

RangeQueriesInfo（RangeQueryInfo）：[

StartKey：StartKey,

EndKey：EndKey,

ItrExhausted：ItrExhausted,

ReadsInfo：

ReadsInfo

],

Writes（KVWrite）：[

Key：Key,

IsDelete：IsDelete,

Value：Value

]

}

]

},

Events（ChaincodeEvent）：{

ChaincodeId：ChaincodeId,

```
                                TxId：TxId,
                                EventName：EventName,
                                Payload：Payload
                            }
                            Response：{
                                Status：Status,
                                Message：Message,
                                Payload：Payload
                            },
                            ChaincodeId：ChaincodeId
                        }
                    },
                    Endorsement：[
                        Endorser：Endorser,
                        Signature：Signature
                    ]
                }
            }
        ]
    }
},
Signature：Signature
}
```

整个信封 Envelope 的 Signature 是交易提交者对整个 Envelope.Payload 的签名。应用程序可以把生成的交易信封内容发送给任意选择的几个排序服务节点。

4.3.5 排序服务节点以对交易进行排序并生成区块

排序服务不读取交易的内容，如果在生成交易信封内容的时候伪造了交易模拟执行的结果，排序服务节点也不会发现，但会在 终的交易验证阶段校验出来并标记为无效交易。

排序服务要做得很简单，先是接收网络中所有通道发出的交易信息，读交易信封的 Envelope.Payload.Header.ChannelHeader.ChannelId 以获取通道名称，按各个通道上交易的接收时间顺序对交易信息进行排序，生成区块。

4.3.6 排序服务节点以广播给组织的主节点

排序服务节点生成区块以后会广播给通道上不同组织的主节点。

4.3.7 记账节点验证区块内容并写入区块

背书节点是动态角色，只要参与交易的背书就是背书节点，哪些交易选择哪些节点作为背书节点是由应用程序选择的，这需要满足背书策略才能生效。所有的背书节点都属于记账节点。所有的 Peer 节点都是记账节点，记录的是节点已加入通道的账本数据。记账节点接收到的是排序服务节点生成的区块，验证区块交易的有效性，提交到本地账本后再产生一个生成区块的事件，监听区块事件的应用程序可以进行后续的处理。如果接收到的区块是配置区块，则会更新缓存的配置信息。记账节点的处理流程如图 4-4 所示。

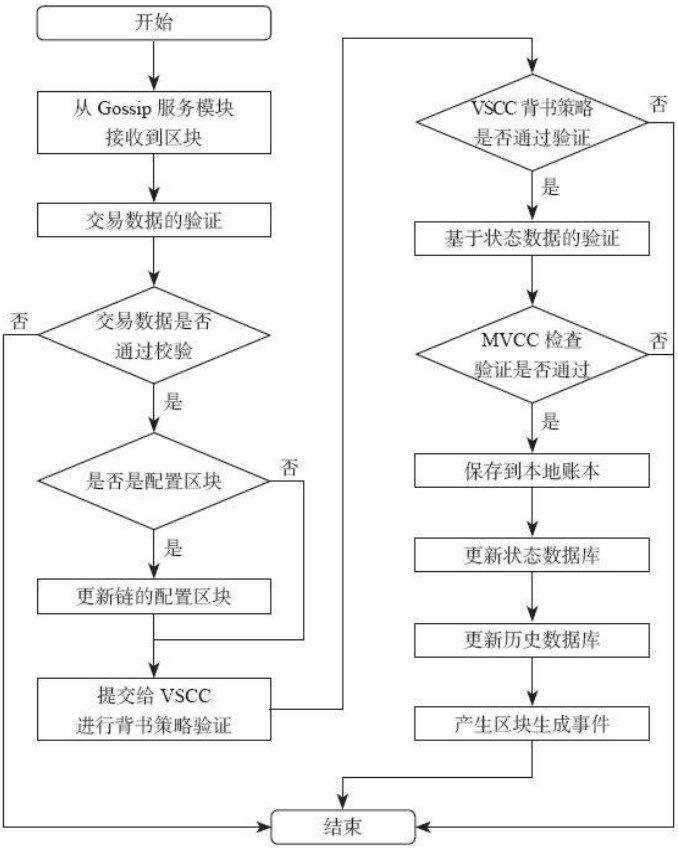

图 4-4: 记账节点的流程图

1、交易数据的验证

区块数据的验证是以交易验证为单位的，每次对区块进行验证时都会生成一个交易号的位图 TxValidationFlags，它记录每个交易号的交易验证状态，只有状态为 TxValidationCode_VALID 才是有效的。位图也会写入到区块的元数据 BlockMetadataIndex_TRANSACTIONS_FILTER 中。交易验证的时候会检查以下内容：

- 是否为合法的交易：交易格式是否正确，是否有合法的签名，交易内容是否被篡改；
- 记账节点是否加入了这个通道。

基本的验证通过以后会提交给 VSCC 进行背书策略的验证。

2、记账节点与 VSCC

链码的交易是隔离的，每个交易的模拟执行结果读写集 TxRwSet 都包含了交易所属的链

码。为了避免错误地更新链码交易数据，在交易提交给系统链码 VSCC 验证交易内容之前，还会对链码进行校验。

3、基于状态数据的验证和 MVCC 检查

交易通过 VSCC 检查以后，就进入记账流程。kvledger 还会对读写集 TxRwSet 进行 MVCC（Multi-Version Concurrency Control）检查。

kvledger 实现的是基于键值对（key-value）的状态数据模型。对状态数据的键有 3 种操作：

- 读状态数据；
- 写状态数据；
- 删除状态数据。

对状态数据的读操作有两种形式：

- 基于单一键的读取；
- 基于键范围的读取。

MVCC 检查只对读数据进行校验，基本逻辑是对模拟执行时状态数据的版本和提交交易时状态数据的版本进行比较。如果数据版本发生变化或者某个键的范围数据发生变化，就说明这段时间之内有别的交易改变了状态数据，当前交易基于原有状态的处理就是有问题的。由于交易提交是并行的，所以在交易未打包生成区块之前，并不能确定 终的执行顺序。如果交易执行的顺序存在依赖，在 MVCC 检查的时候就会出现依赖的状态发生了变化，实际上是数据出现了冲突。图 4-5 所示为基于状态的数据验证的流程图。

写集合本身包含了写和删除的数据，有一个状态位标识是否删除数据。为了提升效率，状态数据库的提交是批处理的，整个区块交易的状态数据同时提交，这也保证了整个区

块的状态数据要么都提交成功，要么都提交失败。这时只会出现记录的账本数据和状态数据库不一致，不会出现区块的状态数据不一致的情况。当账本数据和状态数据库不一致时，可以通过状态数据库的检查点来标记。

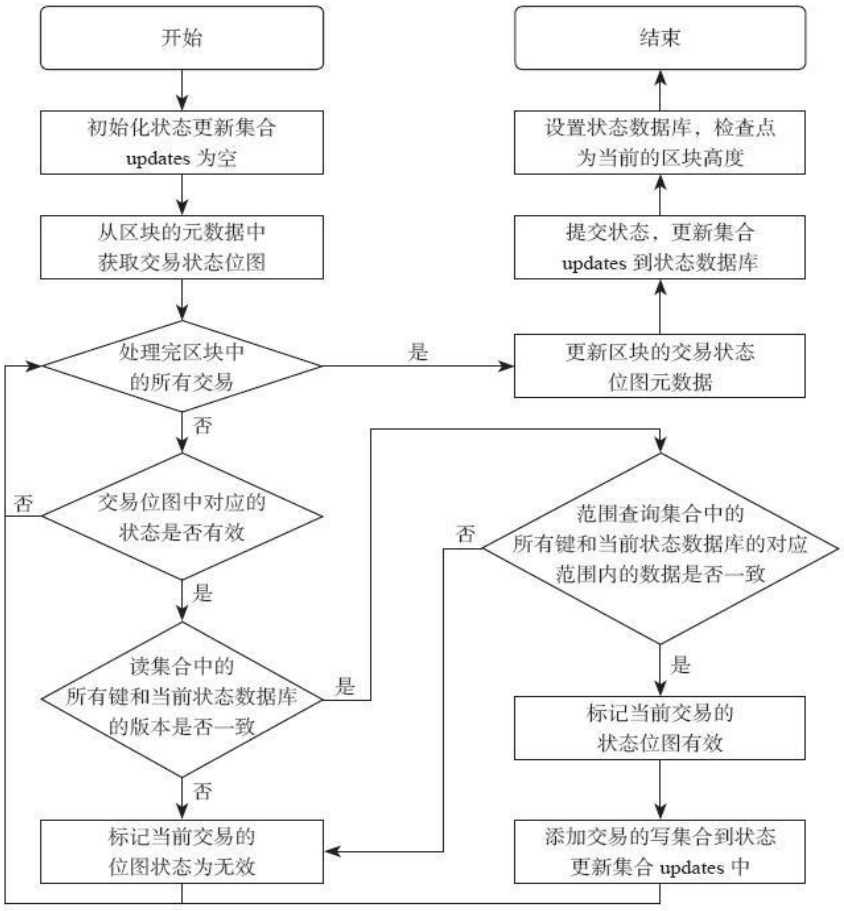

图 4-5： 基于状态的数据验证的流程图

4、无效交易的处理

伪造的交易会导致无效交易，正常的交易也可能出现无效交易。MVCC 检查的是背书节点在模拟执行的时候，环境是否和记账节点提交交易时的环境一致，这里的环境是指状态数据库里数据的三元组（key、value、version）是否完全一致。如果正常提交的交易在这个过程中涉及的数据发生了变化，那么也会出现检查失败从而导致无效交易。在这种情况下，需要在上层的应用程序有一些补偿措施，比如调整交易打包的配置，重新提交失败的交易等。

在目前版本的实现中，无效交易也会保留在区块中，可以通过区块记录的元数据确定哪些是无效交易。无效交易的读写集不会提交到状态数据库中，不会导致状态数据库发生变化，只是会占用区块的大小，占用记账节点的硬盘空间。后续的版本会实现账本的精简，过滤掉无效交易。

4.4 消息协议结构

4.4.1 信封消息结构

信封消息是认证内容中最基本的单元。它由一个消息负载（Payload）和一个签名（Signature）组成。

```
// 信封包含一个带有签名的负载，以便认证该消息
message Envelope {
    // 编组的负载
    bytes payload = 1;
    // 负载头中指定创建者签名
    bytes signature = 2;
}
    // 负载是消息内容（允许签名） message Payload {
    // 负载头部，提供身份验证并防止重放
    Header header = 1;
    // 数据，其编码由头的类型定义
    bytes data = 2;
}
```

负载包含：

1、消息头部。头部带有类型，描述负载的性质以及如何解组数据字段。此外，头部还包含创建者的信息和随机数，以及用来标识时间逻辑窗口的时期信息。只有在两个条件都成立的情况下，Peer 节点才能接受一个信封。

2、数据字段的类型由头部指定。头部消息的组织方式如下所示：

```
    message Header {
    bytes channel_header = 1;
    bytes signature_header = 2;
}
    // 通道头是一个通用的预防重放和身份标识的消息，它包含在一个被签名的负载之中
message ChannelHeader {
    int32 type = 1; // 头类型 0-10000 由 HeaderType 保留和定义
    // 版本指示消息协议版本
    int32 version = 2;
    // 时间戳是发件人创建消息的本地时间
    google.protobuf.Timestamp timestamp = 3;
    // 该消息绑定通道的标识符
    string channel_id = 4;
    // 端到端使用唯一的标识符
    //   -  由较高层设置，如最终用户或 SDK
    //   -  传递给背书节点（将检查唯一性）
    //   -  当消息正确传递时，它将被记账节点检索（也会检查唯一性）
    //   -  存储于账本中
    string tx_id = 5;
    // 时期信息基于区块高度而定义，此字段标识时间的逻辑窗口
    // 只有在两个条件都成立的情况下，对方才接受提案响应
    // 1．消息中指定的时期信息是当前时期
    // 2．该消息在该时期内只看到一次（即没有重放）
    uint64 epoch = 6;
    // 根据消息头类型附加的扩展
    bytes extension = 7;
}
enum HeaderType {
    MESSAGE = 0;                    // 非透明消息
    CONFIG = 1;                     // 通道配置
    CONFIG_UPDATE = 2;              // 通道配置更新
```

```
    ENDORSER_TRANSACTION = 3;        // SDK 提交背书
    ORDERER_TRANSACTION = 4;         // 排序管理内部使用
    DELIVER_SEEK_INFO = 5;           // 指示 Deliver API 查
找信息
    CHAINCODE_PACKAGE = 6;           // 链码打包安装
}
message SignatureHeader {
    // 消息的创建者，链的证书
    // 只能使用一次的任意数字，可用于检测重放攻击
    bytes nonce = 2;
}
```

信封消息结构对于验证负载的签名是必要的。否则，对于大载荷消息，就必须连接所有的载荷再进行签名验证，这往往成本很高。

经过排序后，批量的信封消息交付给记账节点进行验证，通过验证后的数据被记录到账本之中。

4.4.2 配置管理结构

区块链有与之相关的配置，配置设置在创世区块之中，但可能在后续被修改。该配置信息在类型为 CONFIGURATION_TRANSACTION 的信封消息中编码。配置信息本身就是区块的一个单独交易。配置信息交易没有任何依赖，所以每个配置信息交易必须包含对于链的全量数据，而不是增量数据。使用全量数据更容易引导新的 peer 或排序节点，也便于未来进行裁剪工作。

CONFIGURATION_TRANSACTION 类型的信封消息具有 Configuration Envelope 类型的负载数据。

它定义为：

```
    message ConfigurationEnvelope {
    repeated SignedConfigurationItem Items = 3;
}
```

配置信息的信封消息有与之关联的序列号和链 ID。每次修改配置序列号必须递增，这可以作为防止重放攻击的一个简单机制。配置信息的信封中会嵌入一系列的 SignedConfigurationItems，定义如下：

```
    message SignedConfigurationItem {
    bytes ConfigurationItem = 1;
    repeated Envelope Signatures = 2;
}
```

因为 SignedConfigurationItem 必须支持多个签名，所以它包含一组重复的信封消息。这些消息中每个都有一个类型为 CONFIGURATION_ITEM 的头部。负载的数据部分在 ConfigurationItem 中保存，定义为：

```
message ConfigurationItem {
enum ConfigurationType {
        Policy = 0;
        Chain = 1;
        Orderer = 2;
        Fabric = 3;
}
    uint64 LastModified = 2;
    ConfigurationType Type = 3;
    string ModificationPolicy = 4;
    string Key = 5;
    bytes Value = 6;
}
```

Type 提供了配置项的范围和编码信息。LastModified 字段设置为上一次配置项被修改

时配置信息信封中的序列号。ModificationPolicy 指向一个已经命名的策略，用来对将来的签名进行评估，以确定修改是否被授权。Key 和 Value 字段分别用作引用配置项及其内容。

任何有权更改配置项的角色都可以构建新的配置信息交易。修改配置项将更新序列号并产生新的创始区块，这将引导新加入网络的各种节点。

4.4.3 背书流程结构

当 Envelope.data 中携带与链码相关的消息时，使用 ENDORSER_TRANSACTION 类型。

获得批准的 ENDORSER_TRANSACTION 负载流程如下。

首先，客户端向所有相关的背书节点发送提案消息（提案基本上是要进行一些影响账本的动作）。

然后，每个背书节点向客户端发送一个提案响应消息。提案响应包含背书结果的成功/错误码、应答负载和签名。应答负载之中包含提案的哈希值信息，用此信息可以将提案和针对该提案的应答安全地连接起来.

最后，客户端将背书结果写入交易中，签名并发送到排序服务。

1、交易提案结构

一个提案消息包含头部（包含描述它的一些元数据，例如类型、调用者的身份、时间、链的 ID、加密的随机数）和不透明的负载：

```
message SignedProposal {
    // 提案
```

```
    bytes proposal_bytes = 1;
    // 对提案进行签名，该签名将和头部的创建者标识进行验证
    bytes signature = 2;
}
message Proposal {
    // 提案头部
    bytes header = 1;
    // 提案负载，具体结构由头部的类型决定
    bytes payload = 2;
    // 提案的可选扩展。对于 CHAINCODE 类型的消息，其内容可能
是 ChaincodeAction 消息
    bytes extension = 3;
}
```

当背书节点收到签名后的提案消息后，它将验证消息中的签名。验证签名需要以下步骤。

- 预验证用户生成签名证书的有效性。一旦 SignedProposal.proposal_bytes 和 Proposal.header 都解组成功，就可以认为证书基本是可用的。虽然这种在证书验证前的解组操作可能并不太理想，但是在这个阶段可以过滤掉证书过期的情况。
- 验证证书是否可信（证书是否由受信任的 CA 签名），并允许交易（能否通过 ACL 检查）。
- 验证 SignedProposal.proposal_bytes 上的签名是否有效。
- 检测重放攻击。

以下是当 ChainHeader 的类型为 ENDORSER_TRANSACTION 时的消息：

```
message ChaincodeHeaderExtension {
    // 控制提案的负载在最终交易和账本中的可见程度
    bytes payload_visibility = 1;
    // 要定位的链代码 ID
    ChaincodeID chaincode_id = 2;
```

```
}
```

ChaincodeHeaderExtension 消息用于指定要调用的链码以及应在账本中呈现的内容。

理想情况下，payload_visibility 是可配置的，支持至少 3 种主要可见性模式：

- 负载所有字节都可见；
- 只有负载的哈希值可见；
- 任何东西都不可见。

2、提案响应结构

提案响应消息从背书节点返回给提案客户端。背书节点使用该消息表达对于交易提案的处理结果。应答结果可能是成功也可能是失败，另外还会包含动作描述和背书节点的签名。如果足够数量的背书节点同意相同的动作并进行签名，则

可以生成负载消息，并发送给排序节点。

```
message ProposalResponse {
    // 消息协议版本
    int32 version = 1;
    // 消息创建时间，由消息发送者定义
    google.protobuf.Timestamp timestamp = 2;
    // 某个动作的背书是否成功
    Response response = 4;
    // 负载，ProposalResponsePayload 字节序列
    bytes payload = 5;
    // 提案的具体背书内容，基本上就是背书节点的签名
    Endorsement endorsement = 6;
}
message ProposalResponsePayload {
    // 触发此应答交易提案的哈希值
    bytes proposal_hash = 1;
```

```
    // 扩展内容，应该解组为特定类型的消息
    bytes extension = 2;
}
message Endorsement {
    // 背书节点身份（例如，证书信息）
    bytes endorser = 1;
    // 签名，对提案应答负载和背书节点证书这两个内容进行签名
    bytes signature = 2;
}
```

proposal_hash 字段将交易提案和提案响应两者对应起来，即为了实现异步系统的记账功能也为了追责和抗抵赖的安全诉求。哈希值通常会覆盖整个提案消息的所有字节中。但是，这样实现就意味着只有获得完整的提案消息才能验证哈希值的正确性。

出于保密原因，使用链码不太可能将提案的负载直接存储在账本中。例如，类型为 ENDORSER_TRANSACTION 的消息，需要将提案的头部和负载分开进行处理：头部总是进行完整散列的，而负载则可能进行完整散列或对哈希值再进行散列，或者根本不进行散列。

3、背书交易结构

客户端获得足够的背书后，可以将这些背书组合成一个交易信息。这个交易信息可以设置为负载信息的数据字段。以下是在这种情况下要使用的具体消息： message Transaction {

```
    // 负载是一个 TransactionAction 数组，每个交易需要一个数组来适应多个动作
    repeated TransactionAction actions = 1;
}
message TransactionAction {
    // 提案头部
    bytes header = 1;
```

```
    // 负载由头部类型决定，它是 ChaincodeActionPayload 字节序列
    bytes payload = 2;
}
```

ChaincodeEndorsedAction 消息承载有关具体提案的背书信息。

proposalResponsePayload 是由背书节点签名的，对于 ENDORSER_ TRANSACTION 类型，ProposalResponsePayload 的 extenstion 字段会带有一个 ChaincodeAction。此外，endorsements 字段包含提案已经收到的背书信息。

4.5 策略管理和访问控制

在 EOTC 中，较多的地方都使用策略进行管理，它是一种权限管理的方法，包括交易背书策略、链码的实例化策略、通道管理策略等。

4.5.1 交易背书策略

交易背书策略是对交易进行背书的规则，是跟通道和链码相关的，在链码实例化的时候指定。在链码调用的时候，需要从背书节点收集足够的签名背书，只有通过背书策略的交易才是有效的。这是通过应用程序和背书节点之间的交互来完成的，这在前面的交易流程里已经介绍过了。

4.5.2 链码实例化策略

链码实例化策略是用来验证是否有权限进行链码实例化和链码升级的。链码实例化策略是在对链码打包和签名的时候指定的，如果没有指定实例化策略，默认是通道的管理员才能实例化。

```
type SignedChaincodeDeploymentSpec struct {
    // 链码部署规范
    ChaincodeDeploymentSpec []byte
```

```
    // 链码的实例化策略，结构同背书策略，在实例化的时候验证
    InstantiationPolicy []byte
    // 链码所有者的签名背书列表
    OwnerEndorsements []*Endorsement
}
```

链码实例化策略的定义和背书策略完全一样，验证方法也相同，只是用途和用法不一样。链码实例化策略是直接从链码打包中获取的，实例化完成后会将策略存放在链上。在链码实例化和升级的时候会先验证是否符合当前的实例化策略，验证通过才可以更新链码实例化策略。存储在链上的链码信息结构如下所示：

```
type ChaincodeData struct {
    // 链码名称
    Name string
    // 链码版本
    Version string
    // 链码的 ESCC
    Escc string
    // 链码的 VSCC
    Vscc string
    // 链码的背书策略
    Policy []byte
  // 链码的内容：包含链码的名称、版本、链码源码哈希、链码名称和版本的元数据哈希等内容
    // 不包含链码源码
    Data []byte
    // 链码指纹标识，目前没有使用
    Id []byte
    // 链码实例化策略
    InstantiationPolicy []byte
}
```

链码信息结构 ChaincodeData 在链上是按链码的名称索引的。

4.5.3 通道管理策略

通道配置是递归定义的：

```
type ConfigGroup struct {
    Version    uint64                          // 配置版本
    Groups      map[string]*ConfigGroup    // 子配置
    Values      map[string]*ConfigValue    // 配置值
    Policies   map[string]*ConfigPolicy   // 配置策略定义
    ModPolicy string                          // 配置修改策略的名称
}
```

从上面的定义中我们可以看到，配置策略是基于 SignaturePolicy 和 ImplicitMetaPolicy 的，ModPolicy 代表的是修改同级策略用到的策略名称。通道定义了 3 种配置策略，如下表所示。

策略名称	策略含义	策略说明
Readers	通道读取权限	任何读取通道交易的权限控制策略。比如定义哪些身份或者组织可以读取链上的数据，也包括调用排序服务的 Deliver 接口的权限和接收通道内事件的权限
Writers	通道写入权限	任何向通道提交交易的权限控制策略。比如定义哪些身份或者组织可以向链上提交交易，控制可以通过背书节点提交交易提案的权限
Admins	通道管理权限	任何修改通道配置的权限管理策略。比如定义哪些身份或者组织可以有通道配置的管理员权限，它指定了修改通道时需要的管理员签名

以上从逻辑结构、节点结构、典型交易流程、消息协议结构、策略管理等几个方面介绍了 EOTC 架构的设计原则和思路。

5.团队及社区自治组织

5.1 团队介绍

EOTC 团队成员均为社区成员，成员热衷于 EOTC 项目的开发和运营，但出于个人原因，本白皮书中不公布成员的真实姓名。

- A：项目经理

多个区块链项目管理经验，运营过千万级用户的 IT 项目。

- B：系统架构师

py-evm 区块链首席架构师，为以太坊协议 Serenity 规范提供清晰简单的 API 功能。

- C：智能合约研发工程师

丰富的 Solidity 开发经验，早期对 Hyperledger 有较深入的研究

- D：数据分析师

具有分散式 oracle 解决方案的项目经验，曾在 Chainlink 维护 BigQuery 的工作经历。

- E：高级软件工程师

多年开发经验，Golang、Rust 资深工程师，曾开发依托 XCMP 协议的跨链通信功能，polkadot 开源社区贡献者。

- F：安全工程师

在密码学、和智能合约审计方面有着深厚的经验，是去中心化共识研究和安全区块链基础架构领域的专家

5.2 社区自治组织

EOTC 目前已经获得国际知名投资机构 Armlon.Inc、KR19 capital、Limitchain capital、Ripple Ventures Capital、Blockchain Ventures Capitl 等的投资，很快将在全球进行市场的运营推广及布局，在全球各国和地区设立 EOTC-DAO 自治社区组织，打造一个基于区块链的全球社区用户共建自治的生态社区。EOTC-DAO 生态社区，建设社区区块链多态链接及智能化处理装置，利用智能合约和 DAO 解决 EOTC 生态建设和应用落地问题。

在 EOTC-DAO 生态社区，用户可以发起提案和参与提案投票，并通过分布式账本的使用，将投票人的每一票都真实公开地记录在区块链上，真正公平、公正、公开，实现了全网用户共建 EOTC 的生态，助推 EOTC 的生态发展及应用落地。EOTC-DAO 社区的治理规则如下：

1、拥有 EOTC 总量的 1%可以进行提案，提案发起后进入投票期；

2、7 天投票期内，如果 EOTC 供应量的 4% 投赞成票，则提案通过；

3、投票结束后 2 个自然日内开始执行提案。

社区用户的活跃来自产品功能的设计，EOTC-DAO 生态社区将陆续开发一系列的产品功能，服务于社区用户，增强社区用户的粘性，提高社区用户的活跃度。

EOTC-DAO 生态社区还将建立一套先进完整的社区管理系统，让每个社群、社区都能更好地管理自己的用户群体，从而能够更好地服务用户群体。该管理系统将大大提高社区建设的效率和提升社区管理的能力，从而增强社区用户的体验和活跃性。

如何让整个社区更有粘性，如何增加更多的社区用户，如何培养更多的社区创建者和社群节点人，如何保障社区用户的利益，如何解决社区的信任问题，如何让未来更多的区块链落地应用得到有效实施，这些都是 EOTC-DAO 生态社区要致力于解决的问题。

EOTC-DAO 生态社区将通过不懈的努力，结合 DAPP 应用层产品功能和区块链底层技术，致力于打造一个基于区块链的全球社区用户共建自治的生态社区平台。

6. 发行方案

6.1 EOTC 数字资产

EOTC 发行的数字资产为 EOTC，EOTC 是治理 Token，允许 EOTC 社区持有和治理 EOTC 协议。EOTC 不仅可以用于手续费折扣，可以享受协议的所有收益。EOTC 具备多种属性：

- 物权属性：使用权，确定资产归属
- 投资属性：可增值，快速收益
- 货币属性：可流通，在生态系统内是硬通货
- 股权属性：可参与治理，可增值，长期收益，升值空间大

任何一种货币或者商品，其价格都是围绕着价值波动的，短期来看金融市场的价格波动是不理性的，没有监管的币圈的价格波动更为激烈。EOTC 不同于那些靠庄家拉盘百倍千倍再抛给韭菜接盘的山寨币，EOTC 的团队专注于为区块链行业创造价值，提供更优质的交易服务。长期来看，EOTC 代币价值是无限的。

6.2 发行方案

EOTC 总发行量为 2,100,000,000（21 亿枚），数量恒定。发行方案如下：

表 6-1：EOTC 发行方案

	分配明细	占比	数量
社区 39%	社区预留	7%	147,000,000
	社区节点	10%	210,000,000
	奖励空投	2%	42,000,000
	市场	20%	420,000,000
团队 26%	创始人	5%	105,000,000
	前期投资人	4%	84,000,000
	未来员工	7%	147,000,000
	技术团队	6%	126,000,000
	运营团队	4%	84,000,000
区块 35%	区块奖励	25%	525,000,000
	保险基金	5%	105,000,000
	公关基金	5%	105,000,000

注：空投奖励上线两个月后奖励承兑、用户、节点、持币

- 通缩机制：EOTC 定价由市级节点投票通过即可执行，社区全节点投票通过可反对。

手续费 80%分配给社区节点。其中：

- 20%按权重分给高级节点
- 20%按权重分给中级节点
- 20%按权重分给实时节点
- 20%实时奖励给所属信用节点

- 通胀机制：社区代币中的区块奖励发放完后，按每年 2%的通胀进行区块奖励；社区可投票修改通胀率。
- 释放机制：

- 团队 26%：分四年释放，顺序：40%，30%，20%，10%。
- 市场 20%：分 9 个月平均释放。

7. 免责与风险声明

7.1 免责声明

本文档仅提供和项目相关的信息；本文档或文档中的任何内容均不得视为招揽，提议购买，出售任何证券、期货、期权或其他金融工具，或向任何司法管辖区的任何人提供或提供任何投资建议或服务；本文档中的任何内容均不构成投资建议或对任何证券的适用性提供任何意见。过去的表现不一定表示未来的表现，本文档中的任何预测，市场前景或估计均为基于某些假设的前瞻性陈述，不应该被视为指示将发生的实际事件。

意向兑换人若自行决策后进行兑换，应当完全接受该等风险，并愿意自行为此承担一切相应结果或后果。基金会及团队明确表示不承担任何参与 EOTC 项目造成的直接或间接的损失，包括但不限于:

- 因为用户交易操作带来的经济损失；
- 由个人理解产生的任何错误、疏忽或者不准确信息；
- 个人交易各类区块链资产带来的损失及由此导致的任何行为。

7.2 风险声明

EOTC 开发和运营团队相信，在 EOTC 的开发、维护和运营过程中存在一定的风险，部分风险可能会超出团队的控制。除本白皮书所述的其他内容外，每个 EOTC 的参与者还应该细读、理解并仔细考虑下述风险：

- 监管风险：区块链技术目前属于发展阶段，各国对于区块链项目的监管政策存在不确定性，本项目可能会面临运营管理政策方面的风险；

- 信息披露风险：截止到本白皮书发布之日，EOTC 平台仍在不断完善，其哲学理念、共识机制、推演算法和代码以及其他技术细节和参数可能频繁随时发生变化和更新。尽管本白皮书包含了 EOTC 最新的关键信息，但并非绝对完整。且仍会被 EOTC 开发和运营团队为了特定目的不时进行调整和更新。EOTC 开发和运营团队无能力且无义务告知参与者 EOTC 平台在开发中的每个技术细节，因此信息披露的不充分是不可避免且合乎情理的。
- 竞争风险：交易所平台是一个竞争异常激烈的领域，有大量团队正在计划并着手开发，竞争将是残酷的，但在这个时代，任何好的概念，创业公司甚至是成熟的公司都会面临这种竞争的风险。但对我们来讲，这些竞争都是发展过程中的动力。
- 团队风险：EOTC 的团队十分优秀，但是天下没有不散的宴席。EOTC 在发展过程中，团队成员可能会因压力、身体、个人等因素离职。我们会尽力保障团队的完整性和梯队建设。
- 商业风险：EOTC 的团队将全力以赴实现，通过完善的商业模型实现项目的发展目标。但我们同时也意识到市场是复杂的，鉴于行业整体发展趋势存在不可预见因素，现有的商业模型也许会存在与市场需求脱节、从而导致盈利难以达到预期目标的风险。

EOTC 互联网域名：eotc.im；eotc.me

EOTC 以太坊域名：eotc.eth