www.platon.network

PlatON

PlatON：高性能 Trustless 计算网络

V0.6.5

2018 年 11 月 28 日

“世界不仅是一台最值得称道的机器，而且就其心灵组成而言，它也是一个最好的共和国。”

Gottfried Wilhelm Leibniz
——《论万物的终极起源》

序章——云图

我们身处于一个由**复杂性**构筑的时代。

人与人、人与物、物与物之间的关系，“涌现”般地从单向性、单线性的关系进化到多维度的复杂拓扑结构。

从当下开始，直到可见的未来，数以百亿乃至万亿计的智能节点逐步加入并组成全球计算网络。“节点”们不同程度上掌握了充分甚至冗余的算力与存储，期待着与世界的链接与共识。

在农业社会，人类以“观察”获取数据；在工业社会，人类以“测量”获取数据；在信息社会，海量数据遗留在互联网上，人类以“一键记录”的方式获取数据。但数据的主权何属？隐私何归？价值何如？

节点之间正在产生难以计量的多维度的链接，多层次地定义了万事万物的价值、**“链接”**产生了意义，决定了某一“节点”的特定状态对其他“节点”的影响或改变，产生了新的**度量衡**。

这一切正在并将逐步建构真正意义上的复杂网络，呈现为高度自组织的分布式体系。这一基本事实亦将逐步解构乃至瓦解传统计算架构对于治理结构、算法、算力与数据的垄断，直至新一代的全数字化基础设施横空出世。

结构决定功能。目前几乎所有的区块链技术体系都仍局限于节点之间的“信任”与性能问题。但如果继续沿袭这样的思路，我们将仍然无法摆脱今天这个离散、孤立、无隐私保护的互联网架构，我们也将仍然无法处理各类复杂问题。

将复杂性还原为根本问题之后，全数字化世界的公共基础设施可以展开为：数据的流动性、多源异构网络的自组织与可装配的隐私保护。要实现充分的数据交换与协同计算，这一切都源于无所不在的计算。也只有依赖这一不断进化中的基础设施，我们才可以真正展望人工智能可计算的未来。

自莱布尼茨始，计算日益成为科学与哲学的基础方法论，并已经在从原子世界到比特世界中呈现出统一的威力。计算是对数据和信息的处理过程；计算是宇宙与生命存在和进化的基本方式；计算是人类认知和行为的基础范式。

PlatON 是面向未来的下一代计算架构，她的发展与完善是一次软硬件的**协同进化过程**。从生态治理、业务重构、网络运营到应用分发，无不涉及到计算复杂性与通讯复杂性的平衡与突破，承载了计算体系架构的新变革。

PlatON 是一次根植于基本哲学理念的践行，从技术层面展开为计算复杂性领域中各个分支的理论突破、算法进化及工程实践；从业务层面展开为全数字化世界的超级基础设施，是对各个行业传统业务治理和网络的拓扑重构；从社区生态层面将会是人类集体的科学探索与智慧融合；从网络层面 PlatON 将会逐步覆盖空天地一体，面向广域的计算节点组网，先脚踏实地，尔后仰望星空。

基于我们的信念与积累，在历经了两年之久的酝酿与反复验证之后正式推出 PlatON，我们将根据路线图逐一实现和打磨每一个细节。PlatON 是一次对于未来计算架构的展望与实践，是一次对于区块链技术和计算复杂性领域的致敬与超越，致力于为下一个世代的分

布式密码经济体提供共享的、运营商级的服务，更是对人类全数字化时代公共基础设施治理服务的全面阐释，以此拥抱扑面而来的新时代。

本文聚焦于 PlatON 在第一个历史阶段的技术架构、网络架构、计算框架等，给出了相关服务和应用的实现。

目　　录

第 1 章

Trustless 计算

1.1 Trustless 计算概述

在数字化世界，每天都会产生大量的数据，但任何单一实体永远都只能掌握数据集合的局部，而没有任意实体可以实时获取所有的全局数据。这是数字化时代面临的基本挑战——**“盲人摸象”**。

每个数字化世界的参与者都是“盲人”，其拥有的数据不足以反映全量数据—“大象”的特征。合作伙伴需要通过数据交换或者协同计算来共享价值、信息和资产。但是，不信任的合作伙伴倾向于使用“可信任第三方”来归集计算数据，并验证数据的有效性，可信任第三方不可避免地带来可扩展性和隐私性问题。现在方兴未艾的云计算平台就是典型的“可信任第三方”。

随着现代密码技术的发展，特别是区块链技术的发展，提出了一项新的计算范式，我们称之为 Trustless 计算，意味着无需依赖第三方就可验证计算结果的完整性。

1.2 Trustless 计算模型

1.2.1 基于共识的计算

十年前，比特币首次实现基于分布式账本的密码货币，在比特币系统中，互不信任的各参与方能安全地完成交易。以太坊继承了比特币的共识算法，并首次提出超越分布式账本的智能合约，从而实现应用程序的 Trustless 计算。然而，为了保证计算的正确性，每个计算操作都需要经过绝大多数节点的重复处理来验证计算的正确性，导致了区块链体系里效率和可信任之间的内在矛盾。虽然以太坊被称为大规模 Trustless 计算的世界计算机，但由于可扩展性差，以太坊无法满足实际的分布式商业应用。此外，缺乏隐私保护还限制了区块链用于处理私有数据，阻碍了分布式应用的进一步发展。

目前已经有几百万个中心化应用运行在云计算平台上，同样大规模 Trustless 计算也是所有去中心化应用的基础设施。有超过一百个项目致力于开发新的共识协议，为分布式应用程序开发可扩展的区块链，以便应用快速安全地运行，但这个方向被证明是极其复杂和极具挑战性的。在基于共识的计算方案中，存在以下三个主要的共同问题。

1.2.1.1 验证者的两难困境（The Verifier's Dilemma）

新加坡大学的研究者在《Demystifying Incentives in the Consensus Computer》这篇文章中首次提出验证者两难的概念。由于验证者两难困境 **[The Verifier's Dilemma]** 问题，复杂的计算会破坏网络的完整性，并引发恶意攻击。

区块链系统会对出块的矿工给予奖励，而验证者则没有任何收益。这对简单合约来说可以运行得非常好，因为验证者所付出的算力很有限。但是如果涉及到比较复杂的智能合约，验证者可能会花费大量的资源却没有任何奖励。结果是，许多矿工会跳过验证过程，保留他们有限的计算资源用于下一个更合适的区块，从而打开了通往严重安全漏洞的大门。

因验证者两难导致的一种典型攻击是：恶意矿工为了获得挖矿的优势，会发起一系列计算密集型交易到网络中，耗尽其他矿工的计算资源。验证者两难的另一个副作用是矿工会接受无效的交易而不去执行正规的验证流程，因而牺牲网络的完整性。

1.2.1.2 可扩展性的三元悖论（The Scalability Trilemma）

"三元悖论" **[The Scalability Trilemma]** 之于区块链可扩展性是一个巨大的挑战。其中可扩展性、去中心化以及安全性组成此三元，对于一个区块链系统，理论上不可能同时在此三个维度取得最优化，它必须以牺牲其中若干个因素去换取在另外一个领域上的提升。

比特币 (Bitcoin) 和以太坊 (Ethereum) 等区块链的设计注重去中心化和安全性。但是，这是以牺牲可扩展性为代价的，每一笔交易都要由网络中的每一个节点进行处理，这给交易吞吐量带来了根本性的限制：它不能高于单个节点的交易吞吐量。

可扩展性已被公认为是区块链的最大难题，导致区块链不能支持繁重、复杂的计算，严重制约区块链行业的发展。

1.2.1.3 安全性与隐私性的两难困境

链上共识也缺乏对隐私的保护，每一个节点都能够获得完整的数据备份，所有交易数据都是公开和透明的。一方面来讲，它确实保证了每笔交易的安全性。但在另一方面，它也确实对用户造成了很大的隐私性问题。由于对数据隐私的顾虑，用户不能对区块链真正开放数据，这也进一步限制了区块链上分布式应用的发展。

1.2.2 链下计算

鉴于链上共识既有的局限性，业界越来越倾向于达成这样的共识：链上的功能应该是"验证"而不是"计算"，因为"计算"比"验证"要慢得多。因此更合理的方案是把计算转移到链下进行，构建同时满足可扩展性、隐私性和可验证性的 Trustless 计算网络，让相互不信任的主体之间在链下而非链上进行交互计算。

现在也有一些 Trustless 计算扩容方案，如 Oasis、Truebit、Stark 等等。按照验证方式，链下 Trustless 计算可分为可信硬件、交互博弈、非交互证明，PlatON 完全基于密码算法，采用 VC 算法叠加同态加密和安全多方计算，实现可扩展的、隐私的、可验证的 Trustless 计算。

1.2.2.1 可信硬件（Trusted Hardware）

这类方案以可信硬件（如 Intel 的 SGX）为底层硬件，建立可信执行环境 (Trusted Execution Environment，简称 TEE)，用于执行智能合约。可信硬件本身就是一个第三方，其本质上是一个存在安全边界的隔离的安全飞地，安全边界内代码和数据是解密的，这个安全边界本身就是安全隐患。有些可信硬件如 SGX 也提供远程证明协议，证明代码是运行在可信硬件上，这个证明是基于私钥签名技术。实际上，目前已经有研究者针对英特尔处理器实现了一种推测性执行攻击 **[Foreshadow]**，宣称可以读取受 SGX 保护的内存以及提取 SGX 的私有证明密钥，并且由于 SGX 的证明报告没有跟具体的 SGX 硬件关联，因此只需一台受损的 SGX 就可以侵蚀整个 SGX 生态系统。

1.2.2.2 交互博弈（Interactive Game）

在以太坊上，由于链上计算昂贵，复杂的程序在上面执行非常受限。Truebit 引入交互博弈协议来强制链下节点正确执行计算。解决计算任务的求解者（Solver）一旦受到挑战，必须与验证者一起进行交互验证博弈。通过几轮交互缩小到计算过程中的问题步骤，并最终以最少的计算和费用来解决链上的分歧。求解者与验证者之间的动态交互增加了 Truebit 系统的不确定性并且交互机制意味着长延迟的终结。

1.2.2.3 非交互证明（Non-interactive Proof）

非交互的链下验证计算中，证明者无需同验证者进行信息交互，也不需要指定特定的验证者进行验证，做到公共可验证（publicly verifiable），即任何人可以本地验证计算的正确性。密码学中的可验证计算（Verifiable Computation）允许计算能力受限的客户端将函数计算外包给计算能力更为强大的一方，同时客户端可以在消耗很小资源的前提下验证计算的正确性。

SNARK 和 STARK 是目前两类 VC 最著名的实现，两者均提供零知识证明、保护证明者的隐私以及减少证明的大小和验证时间的性质，因此在某些应用中很实用。SNARK 具有非常短的证明和快速的证明验证性质，但它需要一个复杂的可信初始化过程，这是部署时最大的障碍。STARK 与 SNARK 的最大的区别在于透明性（Transparent），即不需要可信初始化过程。尽管 STARK 是一个非常令人惊艳的构造，但诸如证明生成时间、证明的大小等这些具体效率问题使得它针对实际应用时不是足够实用。

在基于 SNARK/STARK 的方案中，如果要保护数据隐私，则生成证明的过程必须由输入拥有方来执行，因为只有拥有 witness 的证明者才能生成证明。但是，SNARK/STARK 无法保证将计算委托之后输入的隐私性，因为一旦将计算委托给（可能是不可信的）另一方，这类技术只能保证执行证明过程方的隐私，而无法保证原始计算中输入的隐私。

PlatON 而是以在高效的 VC 算法之上叠加同态加密以及安全多方计算的方式，为 Trustless 计算提供完备的解决方案，同时全流程保护用户数据隐私。

1.2.2.4 PlatON Trustless 计算模型

PlatON 实现的高效 VC 算法不需要可信的初始化过程，与之前的 VC 或 Zero-Knowledge Proof（ZKP）解决方案相比，大大减小了证据大小，并加速了证明生成和验证过程。并且 PlatON 架构中，通过叠加全同态加密（FHE）和安全多方计算（MPC），实现真正的隐私计算，保证输入数据以及计算逻辑本身的隐私。与依赖第三方制造商提供的可信硬件或 TEE（例如 SGX）进行计算完整性的可信计算相比，PlatON 上的 Trustless 计算仅依赖于可证伪的密码学假设，从而在其生命周期内提供前所未有的私有数据安全性，而无需信任边界。

PlatON 计算模型独立于底层公链，不受区块链三元悖论的制约，在保证安全性、隐私性的前提下依然能获得较好的可扩展性和去中心化。PlatON 通过电路并行化的拆分，可将 PlatON 的计算单元细分到电路门，并将计算单元随机分发给不同的计算节点并行计算，极大提高计算的水平扩展性和去中心化。同时，通过可验证计算，合约与计算的执行只需要进行一次，所有节点可以快速验证计算的正确性，提高单个交易的处理性能，交易吞吐量也得到相应的提升。

当将计算外包给其他节点时，传统的 zk-SNARK/STARK 技术无法解决输入隐私问题。这就是为什么我们可以使用确定性证明者/计算者过程来构建 VC，而不是使用 SNARK(zk-SNARK) 的完整机制。我们要强调的是，所有目前已知的 zk-SNARK 构造依赖于一些特殊的和非标准的复杂性假设，例如指数知识假设（knowledge-of-exponent assumption）的更强变体，并且它们通常很复杂且难以解释。这些新的假设是否经得起时间的考验仍是未知问题。

作为众所周知的 VC 方案，SNARK 可为任何 NP 语句生成恒定大小的证明，并具有极快的验证时间。然而，这种系统的证明者/计算者需要花费准线性时间 $O(nlogn)$ 来生成证明。在我们的 VC 中，虽然证明稍微大一些（目标函数大小的对数，与 STARK 相比渐近复杂度一致，但具体长度更短），但它接纳更高效的计算者（线性时间内生成证明），在我们的场景中，计算者的这种较低时间复杂度是更理想的。未来，我们将探索并行化计算的可能性，并使证明生成更有效。

第 2 章

PlatON 技术战略

图2.1红色虚框内为 PlatON，PlatON 实现一个去中心化系统的基础设施层和一条主链。首先，PlatON 定位于给所有去中心化系统（包括区块链、分布式人工智能、科学计算等）

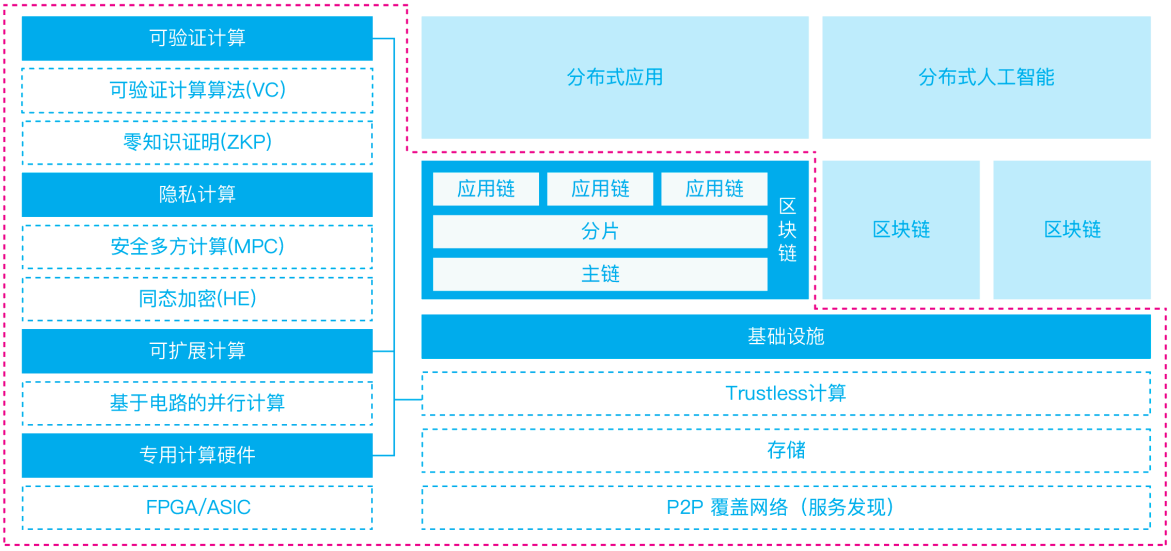

图 2.1　PlatON 技术定位与目标

提供基础设施（包括网络、存储、计算等）。初期目标是完全基于密码学算法构建可扩展 Trustless 计算网络，从根本上解决区块链的可扩展性、隐私性和可验证性问题。

- PlatON 实现高扩展的链下可验证计算，支持繁重、复杂的计算密集型任务，同时采用可验证计算算法来实现数学可证明的计算正确性。
- PlatON 通过安全多方计算和同态加密算法实现真正的链下隐私计算，整个计算的全流程都能保证数据隐私，不存在安全边界的隐患。
- 计算可拆分到电路门的粒度，并以电路门的粒度分散到不同的计算节点，一方面可以提供更高的计算并行度，获得更高的可扩展性，另一方面也进一步将权力分散，获得更好的去中心化特性。

- PlatON 的基于电路的计算模式天然与 FPGA 的架构相匹配。

  通过开发基于 FPGA/ASIC 的专用计算硬件，能够极大的提高计算性能，降低功耗/成本。其次，基于底层可扩展 Trustless 计算网络，PlatON 同时也实现自己的主链。
- PlatON 以 Sharding 方式扩展多个应用链。各条链业务相互独立。
- PlatON Trustless 计算网络的去中心化的底层设计，决定了其发展将会很大程度上依赖算力提供者。PlatON 主链及应用链上的 Energon 是一种 Utility，用于算力的度量和清结算。分布式人工智能和科学计算等大型复杂的计算也能通过 PlatON 主链及应用链获取算力。
- 开发者可在 PlatON 主链及应用链上发布智能合约，并对外提供各类数据服务，Energon 也用于数据服务的清结算。

第 3 章

PlatON 技术方案

3.1 PlatON 计算

3.1.1 可验证计算

PlatON 采用密码学中的可验证计算算法来保证链下计算的可靠性。

公共可验证计算（VC）方案允许计算资源受限的用户将对于输入为 u 的函数 F 的计算外包给计算方。然后用户可以验证返回的计算结果 $F(u)$ 的正确性，并且只需执行比函数计算过程更少的工作。

更正式地，公共可验证计算可定义如下。

定义：一个公共可验证计算方案包含了以下定义的三个多项式时间算法：

- 随机密钥生成算法 $KeyGen(F, \lambda)$ 将函数 F 和安全参数 λ 作为输入。它输出一个公开计算密钥 EK_F 和一个公开验证密钥 VK_F；
- 确定性计算算法 $Compute(EK_F, u)$ 将计算密钥 EK_F 和输入 u 作为输入。它输出结果 y（预期为 $F(u)$）和一个 y 的计算正确性证明 π_y；
- 验证算法 $Verify(VK_F, u, y, \pi_y)$ 将验证密钥 VK_F、输入 u、预期结果 y 和证明 π_y 作为输入，如果 $y = F(u)$，则输出 1；否则输出 0。

公共可验证计算（VC）需满足以下性质：

- **正确性**：对于任何函数 F 和任何输入 u，如果 y 为预期计算的结果，则验证算法 $Verify$ 将接受证明。
- **不可伪造性**：对于任何函数 F 和任何概率多项式时间敌手，若 $y \neq F(u)$，那么产生可接受的证明 π 在计算上是不可行的。
- **有效性**：验证算法 $Verify$ 的复杂度或运行时间需要比直接计算 F 的开销要小很多。

在 PlatON 中，我们的 VC 方案取得了另外两个显著的特性：

- **计算有效性** 针对一些函数，生成证明是相对有效且低成本的。
- 不涉及任何信任初始化过程。

当前最先进的构造 VC 算法的模式为：将计算函数 F 转换为算术电路，然后将电路表示为 Hadamad 乘积关系及其线性约束。

更具体地说，任何乘法输入门都有三条线; $left$、$right$ 作为输入线，$output$ 作为输出

线。在该关系中，将每个乘法门的左输入、右输入和输出编码到 $\mathbb{Z}_p$ 上的向量 a_L、a_R、a_O，且满足关系 $a_L \circ a_R = a_O$，这里 $\circ$ 是成对乘法。其他加法门和常数门将被编码到矩阵 W_L、W_R、W_O 以及 $\mathbb{Z}_p$ 上的向量 c 中。同时提供额外约束条件如下形式：

$$W_L \cdot a_L + W_R \cdot a_R + W_O \cdot a_O = c.$$

根据 Hadamad 乘积与离散对数困难问题假设下的约束之间的关系，计算者将生成一个短证明。用户可以非常有效地验证此证明。VC 方案的更多细节叙述如下。

算法 $KeyGen(F, \lambda)$ 以计算目标函数的一个电路 F 和安全参数 λ 为输入，输出向量 $g, h \in \mathbb{G}^n, W_L, W_R, W_O, c$ 以及一个加密散列函数，这些用来作为计算和验证的密钥。

在算法 $Compute(EK_F, u)$ 中，计算者首先计算出针对一个给定输入的函数 F，然后以下面的方式生成一个正确性证明。计算者中从电路的结构中计算出向量 a_L, a_R, a_O，接着计算中使用公共可知的哈希函数生成随机值 y, z，然后基于 y, z 各自计算两个向量 Y, Z。而非直接证明两者间的关系，计算者证明下面的关系 (3.1):

$$\begin{aligned} \langle a_L, a_R \circ Y \rangle - \langle a_O, Y \rangle + \langle Z, W_L \cdot a_L + W_R \cdot a_R + W_O \cdot a_0 \rangle \\ + k(y, z) = \langle Z, c \rangle + k(y, z) \end{aligned} \tag{3.1}$$

这里 $k(y, z)$ 是一个能被两方均可计算出的多项式。计算中生成两个多项式：

$$\begin{aligned} l(x) &= a_L \cdot x + a_O \cdot x^2 + Y^{-1} \circ (Z \cdot W_R) \\ r(x) &= Y \circ a_R \cdot x - Y + Z \cdot (W_L \cdot x + W_O) \end{aligned}$$

并且将 (3.1) 的左边部分编码进内积多项式 $t(x) = \langle l(x), r(x) \rangle$ 的第二个系数中，将右边部分留给用户来计算。计算者从 $\mathbb{Z}_p^*$ 中随机生成一个统一值 s，并使用公共可知的哈希函数来计算向量 $l = l(s), r = r(s)$，以及 $t = \langle l, r \rangle$。最后，计算者将在证明 $\pi = (a_L, a_R, a_O, t_1, t_3, l, r)$ 发送给用户。

在 $Verify(VK_F, u, y, \pi_y)$ 算法中，一旦收到证明，用户首先使用哈希函数获得 y, z，接着得到向量 Y, Z 以及多项式 $k(y, z)$，然后从哈希函数计算出随机值 s。最后用户通过验证下面的条件来检验关系 (3.1) 的证明：

$$\begin{aligned} l &= a_L \cdot s + a_O \cdot s^2 + Y^{-1} \circ (Z \cdot W_R) \cdot s \\ r &= Y \circ a_R \cdot s - Y + Z \cdot (W_L \cdot s + W_O) \\ \langle l, r \rangle &= t_1 \cdot s + (\langle Z, c \rangle + k(y, z)) \cdot s^2 + t_3 \cdot s^3 \end{aligned}$$

为了减少证明的大小，计算者能够将向量 a_L, a_R, a_O, l, r 存储进三个群元素中：

$$A_I = g^{a_L} h^{a_R} \in \mathbb{G},\ A_O = g^{a_O} \in \mathbb{G},\ P = g^l h'^r \in \mathbb{G}$$

这里 h' 可以从 h 计算出来，并且 $P = A_I^s \cdot A_O^{s^2} \cdot h'^{-Y} \cdot w_l^s \cdot w_r^s \cdot w_o$，其中

$$w_l = h'^{Z \cdot W_L},\ w_r = g^{Y^{-1} \circ (Z \cdot W_R)},\ w_o = h'^{Z \cdot W_O}$$

之后，计算者与用户执行一个内积参数系统（可使用 Fiat-Shamir 启发以一个短的非交互证明来实现），其中计算者证明下面的关系：

$$\{(g, h' \in \mathbb{G}^n, P \in \mathbb{G}, t \in \mathbb{Z}_p;\ l, r \in \mathbb{Z}_p^n) :\ P = g^l h'^r \wedge t = \langle l, r \rangle\}$$

在 PlatON 中，利用 VC 结合计算通道来保证计算完整性。

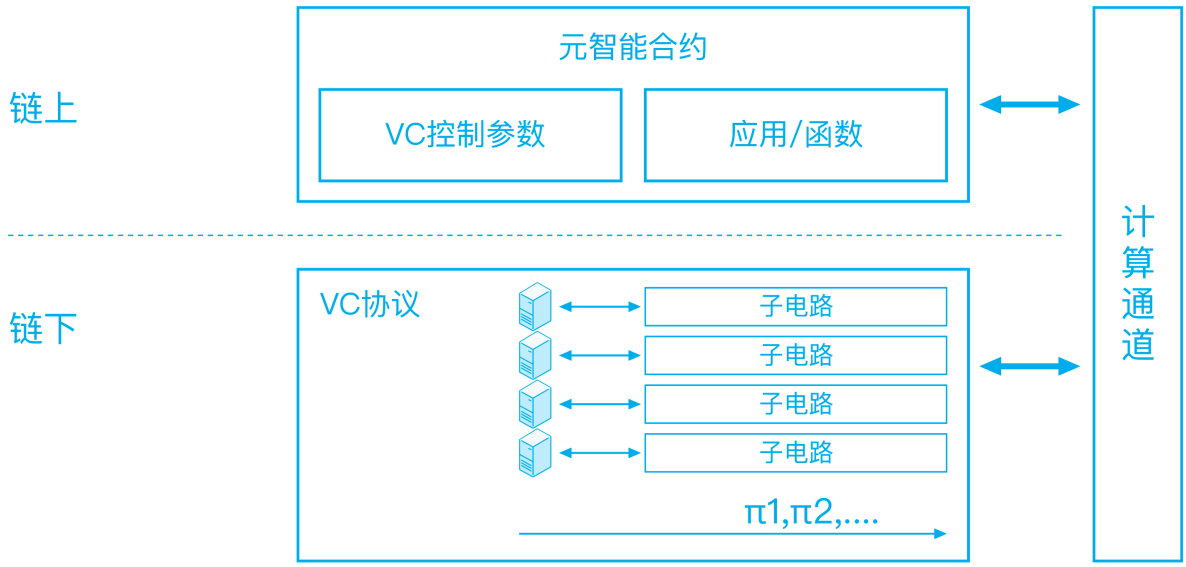

图 3.1　PlatON 中的可验证计算

如图3.1所示，用户以合约的方式发布计算任务，合约中包含 VC 算法的所用到的必要的参数、电路拆分的方式以及其他相应的经济激励。元智能合约的剩余部分为具体的计算任务或者应用。

链上的元智能合约通过计算通道扩展到链下。通过编译器将元智能合约中的计算/应用部分编译成电路，然后在将电路按要求拆分成多个子电路，分发给多个计算节点，计算节点执行子电路，已达到并行计算的效果。为了保证计算节点计算的正确性，其在计算的同时，需要按照 VC 算法生成相应的证明。在提交计算结果后，对 VC 算法生成的证明进行密码学计算，以检查计算输出的正确性。由于 VC 的特性，验证所花费的时间和成本必须远低于原始计算，再加上 VC 研究的最新进展显著地降低了计算节点的开销，链下 VC 总体上比基于链上的计算共识更具可扩展性且消耗更少资源。

PlatON 通过合约计算化，将计算/智能合约编译为电路。对于过于复杂的计算，以电路的形式分拆成多个子任务，再结合计算合约化，通过激励机制吸引网络中的闲置算力来进行子任务的计算。通过可验证计算技术，以极小的算力代价验证异构算力提供的子任务运

算结果。可验证计算连接“合约计算化”与“计算合约化”，最终真正实现利用全球异构算力进行并行计算。

与传统的区块链技术不同，PlatON 中每个节点无需重复运行智能合约来验证交易。利用可验证计算以及电路化的智能合约，节点只需在极短的时间验证交易的合法性，即验证新的状态是否由旧的状态通过智能合约运算而来。

3.1.2 隐私计算

PlatON 通过安全多方计算和同态加密算法实现真正的隐私计算，实现对计算代码和数据的隐私保护。与其他基于 TEE/SGX 的方案不同，PlatON 全流程保证安全，不存在任何安全边界。

3.1.2.1 安全多方计算

安全多方计算（MPC）由姚期智在 1982 年正式提出。它主要探讨的是，n 个参与方各自输入信息去计算一个既定的函数，在保证计算的正确性的同时，不泄露参与方输入数据的隐私。具体来说，对于 n 个参与方，每个参与方 i 均知道自己的输入 x_i，他们想协同计算一个既定函数 $f(x_1, ..., x_n) = y$，使得所有参与方都能获得最终的结果 y，但无法获知其他参与方的输入数据。

构建通用 MPC 协议的最典型范式之一是基于姚期智提出的针对两方计算的加密电路方法，并由 Beaver、Micali 和 Rogaway 进一步扩展到多方计算。

加密电路技术的更多细节在下面描述。对于通用结构，首先将函数 f 表示为布尔电路，不失一般性，由 AND 和 XOR 门组成。由于用高级语言编写的大多数实际程序都包含复杂的数据结构，因此这是一项非常重要的任务。

以两方计算为例，它涉及到两个参与方，称为生成器和计算器。生成器以电路作为输入，写下每个门的真值表。对于电路中的每根线 i，生成器选择两个称为标签的随机字符串 (X_i^0, X_i^1) 来表示 0 和 1。

之后，生成器将真值表中的每个门 g 转换为加密门。定义门 g 的函数为 $g(.,.) : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$。以 AND 门为例，函数 g_{AND} 定义为 $g_{AND}(0,0) = 0$，$g_{AND}(0,1) = 0$，$g_{AND}(1,0) = 0$，$g_{AND}(1,1) = 1$。令 x, y, z 分别为左输入线、右输入线和输出线。生成器计算出每个门 g 的四个密文如下：

$$
\begin{aligned}
c_0 &= H(X_x^0 \| X_y^0 \| g) \oplus X_z^{g(0,0)} \\
c_1 &= H(X_x^0 \| X_y^1 \| g) \oplus X_z^{g(0,1)} \\
c_2 &= H(X_x^1 \| X_y^0 \| g) \oplus X_z^{g(1,0)} \\
c_3 &= H(X_x^1 \| X_y^1 \| g) \oplus X_z^{g(1,1)}
\end{aligned}
\tag{3.2}
$$

这里 H 是一个加密散列函数，并且当 g 作为散列函数的输入时，表示为一个门序列字符串。

在对电路中的所有门进行加密之后，生成器将所有加密密文和与其输入比特串相关的

输入标签（即，0 表示 X_i^0，1 表示 X_i^1）一起发送给计算器。在计算器执行加密电路之前，首先他根据自己的输入信息获得输入标签。由于标签是由生成器提取出来的，因此计算器将和生成器之间执行一个不经意传输协议，在无需告诉生成器明文信息的情况下获得标签。不经意传输是一个非常基础的密码学原语，它满足以下性质：

1. 生成器以标签 X_i^0、X_i^1 为输入，计算器以比特 $b = 0/1$ 为输入，协议执行完成，计算器获得标签 X_b；
2. 生成器不知道计算器选择的是哪个标签；
3. 计算器不知道另一个标签 X_{1-b}。

对于每个加密门，计算器尝试用输入标签作为关键字解密所有四个密文，注意它只能获得一个有意义的输出标签，然后迭代地解密加密门以获得电路的输出标签。由于电路的输出标签的含义是由生成器在开始时揭示，因此计算器可以获得计算的输出，并将其发送回生成器。

学术界提出了许多优化版本的加密电路。包括 Free-XOR 技术，这意味着 XOR 门（几乎）无需加密，row reduction 和 half gate 技术，将每个 AND 门所需要的密文从 4 个降为 2 个。最近使用认证加密电路的研究结果实现了针对两方和多方恶意敌手模型下非常有效的协议。

一个常见的用例是一方拥有算法 A 而另一方拥有输入数据 x，他们希望协同计算 $A(x)$ 而不泄露 A 和 x 的其他信息。在这种情况下，首先应该生成一个以 A 和 x 作为输入的通用电路 U。此通用电路满足 $U(A, x) = A(x)$，这里将 A 表示为一个比特串。

加密电路是安全多方计算的最有效技术之一。最先进的实现能够在个人计算机中每秒处理数百万个门，这对于许多应用来说已经足够有效。

安全多方计算为隐私数据的协同计算提供了根本性的技术手段，在 PlatON 中结合元智能合约的控制层和计算通道提供的经济属性，为具有多源数据输入的应用提供隐私保护，实现真正的隐私计算。

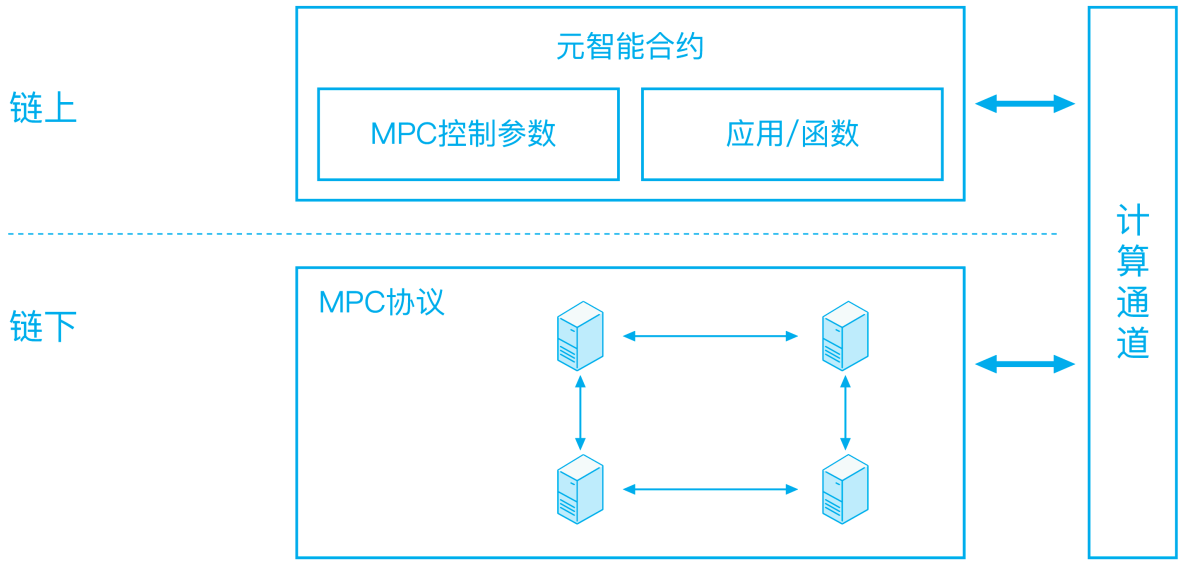

图 3.2　PlatON 中的安全多方计算

如图3.2所示，用户以智能合约的形式广播计算任务，其中一部分合约中包含 MPC 算法的必要参数和涉及到的经济激励，另一部分包含具体的应用或者计算任务。通过计算通道将计算扩展到链下。首先通过编译器将计算/应用部分编译成电路，分发给同时执行 MPC 协议的数据提供方，因此保证了数据留在本地，并且只有（加密的）结果上链。

3.1.2.2 同态加密

同态加密是一种允许在密文上进行计算的加密方式。除了传统加密方案的原始组件之外，还有另一种计算算法，它将目标函数 F 和加密数据作为输入。同态加密会生成一个加密的结果，当解密此结果时，获得的消息就像是在加密数据的明文上执行 F。支持密文上的任意计算的密码系统称为全同态加密（FHE）。正式地说，同态加密包括以下算法:

- 密钥生成算法 $KeyGen(\lambda)$ 将安全参数 λ 作为输入，输出一个公钥 PK 和一个私钥 SK；
- 加密算法 $Enc(PK, m)$ 将公钥 PK 和消息 m 作为输入，输出一个密文 c；
- 解密算法 $Dec(SK, c)$ 将私钥 SK 和密文 c 作为输入，输出消息 m；
- 求解算法 $Eval(PK, f, c_1, \ldots, c_l)$ 将公钥 PK、函数 f 以及密文 $c_1, \ldots, c_l$ 作为输入，输出一个密文 c_f。

除了传统公钥加密方案的性质外，它还满足下面的性质：如果 $c_1 = Enc(PK, m_1)$，...，$c_l = Enc(PK, m_l)$ 以及 $m_f = f(m_1, ..., m_l)$，并且 $c_f = Eval(PK, f, c_1, \ldots, c_l)$，那么则有 $m_f = Dec(SK, c_f)$。

如果对任意函数 f，加密满足以上性质，则称为全同态加密。

现有的 FHE 方案遵循 Craig Gentry 所开创的框架蓝图。Gentry 令人惊讶的开创定理表明，如果一个方案足够同态来计算自己的解密电路，那么它就能够转化为可以计算任何函数的全同态加密。

为了构建一个满足开创定理条件的同态加密，最先进的结构是基于（环上）误差学习（LWE）问题。更具体点，n 为 2 的乘幂，定义一个分圆环 $R = \mathbb{Z}[X]/(X^n + 1)$，对于整数 $q \geq 2$，取 $R_q = R/qR$，定义 R_2 为消息空间。公钥为环 R_q^2 中的一对元素 (a, b)，这里 $b = -a \cdot s - 2e$，密钥为 $\vec{s} = (s, -1) \in R^2$，$s$、$e$ 是从离散高斯分布中提取出来的。

给出一个消息 $m \in R_2$，密文 $\vec{c} = (c_0, c_1)$，这里 $c_0 = a \cdot r + 2e_0$，$c_1 = b \cdot r + 2e_1 + m$，$r$、$e_0$、$e_1$ 是从离散高斯分布中提取出来的。

密文以如下方式解密出来：

$$m = [[\langle \vec{c}, \vec{s} \rangle]_q]_2,$$

这里 $[\cdot]_q$ 意味着对于 $q \geq 2$ 所有运算都是在 $\mathbb{Z}_p$ 中完成。注意只要 $m + 2e_1 - 2e_0 \cdot s$ 的范数相比于 q 足够小，就有

$$[\langle \vec{c}, \vec{s} \rangle]_q = [m + 2e_1 - 2e_0 \cdot s]_q = m + 2e_1 - 2e_0 \cdot s$$

并且解密等式成立。

为了处理密文的同态计算，首先应该将函数 f 表示为算术电路或布尔电路。不失一般性，这里使用 R_2 上的算术电路。因此，只需考虑 R_2 上的加法门和乘法门。

给定两个密文 $\vec{c}_0$、$\vec{c}_1$，要计算加法门，只需进行向量相加，即 $\vec{c}_{add} = \vec{c}_0 + \vec{c}_1 \in R_q$。这

是由于

$$\langle \vec{c}_{add}, \vec{s} \rangle = \langle \vec{c}_0 + \vec{c}_1, \vec{s} \rangle = \langle \vec{c}_0, \vec{s} \rangle + \langle \vec{c}_1, \vec{s} \rangle$$

要计算乘法门，更具有技巧性。注意到对密文 $\vec{c}$，在“误差”为 e' 的情况下，应满足 $[\langle \vec{c}, \vec{s} \rangle]_q = m + 2e'$。从而对于某些“误差” e'' 有：

$$\langle \vec{c}_0 \otimes \vec{c}_1, \vec{s} \otimes \vec{s} \rangle = \langle \vec{c}_0, \vec{s} \rangle \cdot \langle \vec{c}_1, \vec{s} \rangle = (m_0 + 2e'_0) \cdot (m_1 + 2e'_1) = [m_0 m_1 + 2e'']_q$$

这里 $\otimes$ 为张量积。这意味着 $\vec{c}_0 \otimes \vec{c}_1$ 是在私钥 $\vec{s} \otimes \vec{s}$ 下消息 $m_0 m_1$ 的“密文”。为了缩短此“密文”的长度并将其转换为一个“正常”的密文，用到了一个密钥交换技术。基本上地，密钥交换技术将在私钥 $\vec{s} \otimes \vec{s}$ 下的密文 $\vec{c}_0 \otimes \vec{c}_1$ 转换成一个私钥 $\vec{s}$ 下的具有相同明文的密文 $\vec{c}_{mult}$。

在 (R)LWE 结构中，误差管理是做同态运算最重要的问题。几次运算之后（特别是对于乘法门），误差将不断增加，并不断接近 q，从而导致无法解密。为了解决这个问题，另一个称为模数转换（Modulus Switching）的方法被提出来了。简单地讲，它将模数 q 下的密文 $\vec{c}$ 转换为模数 q' 下的具有相同明文的新密文 $\vec{c}'$，并且 $q' < q$。这可简单地由 $\vec{c}' = \left\lfloor \frac{q'}{q} \cdot \vec{c} \right\rceil$ 获得。使用更小的模数会使解密电路更简单并且也能够使用模数交换技术同态地计算出此解密电路。

在 PlatON 中，同样提供单数据源应用的隐私保护。利用同态加密，元智能合约中的控制层和计算通道的经济数据，可达到链下真正的隐私计算。

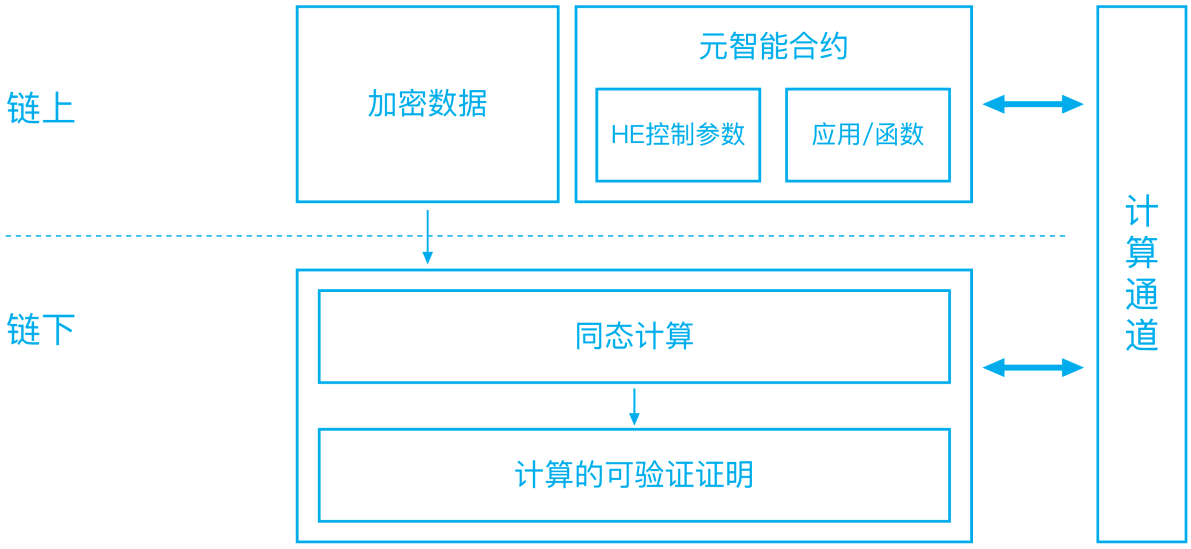

图 3.3　PlatON 中的同态加密

图3.3描述了 PlatON 中的同态加密流程。用户将两部分的信息写进智能合约，第一部分包含 HE 控制参数和激励机制，另一部分描述计算任务。智能合约中的信息一旦通过计算通道转移到链下，立即会被编译成电路，并和由数据提供者通过 FHE 加密后的数据一起以子电路的形式分发给多个计算节点或者单个计算节点执行。VC 协议必须在加密结果上链

之前执行以确保计算正确性。

3.1.3 并行计算

PlatON 中，智能合约被编译成布尔电路（Boolean Circuit），布尔电路是由各种不同的门（Gate）构成的“复杂有向无环图”，可分解为细粒度的计算任务，并通过 PlatON 网络将计算任务分发到多个计算节点并行计算。

为保证计算的可靠性，避免因节点掉线或超时导致计算失败，同一个子任务会同时分发给多个计算节点，保留一定的计算冗余度。

3.2 PlatON 中的电路

电路是由各种不同的门（Gate）通过输入输出线构成的“复杂有向无环网络”。由逻辑门（比如：与、或、非、异或等）构成的电路称为布尔电路（Boolean Circuit）；由算术门（比如加法、乘法等）构成的电路称为算术电路（Arithmetic Circuit）。

任意形式的计算都可由电路表示，电路以有限种类的门构成各类复杂的计算形态。电路因为其基本组成部分的简易性，是在密码学中被广泛使用的计算模型。

PlatON 通过电路来水平地连接各类算法和硬件。电路作为安全多方计算、零知识证明、可验证计算、全同态加密共同使用的通用计算模型，以其超强的普适性串联各类算法。电路表示的算法也天然适合专用硬件的实现。

电路是 PlatON 度量“计算”的基础。构成电路的基本单位为门，不同种类门的资源消耗不同，整个计算的度量可表示为电路中所有门的消耗的度量总和。电路为计算的度量和定价提供了理论基础。

3.3 专用计算硬件

PlatON 中，智能合约的计算逻辑被编译成布尔电路进行计算，整个计算回归到与、非、异或等处理。而布尔电路的操作，天然与 FPGA 的架构相匹配，通过将智能合约转换成 FPGA 的布尔电路并通过 FPGA 来执行这些逻辑单元，能够极大地提高运算效率和降低功耗/成本。

PlatON 将在适当的阶段推出基于 FPGA/ASIC 的专用计算硬件，会极大提升整个区块链平台的交易性能，真正实践下一代计算架构当中的硬件部分。

3.4 PlatON 主链

3.4.1 共识与计算解耦

如图3.4所示，PlatON 将交易执行跟区块链共识解耦，在链下构建可扩展 Trustless 计算网络。一方面将区块链权力进一步分散去中心化实现更高的安全性，另一方面这两个功能可独立升级扩容，获得更好的扩展性。

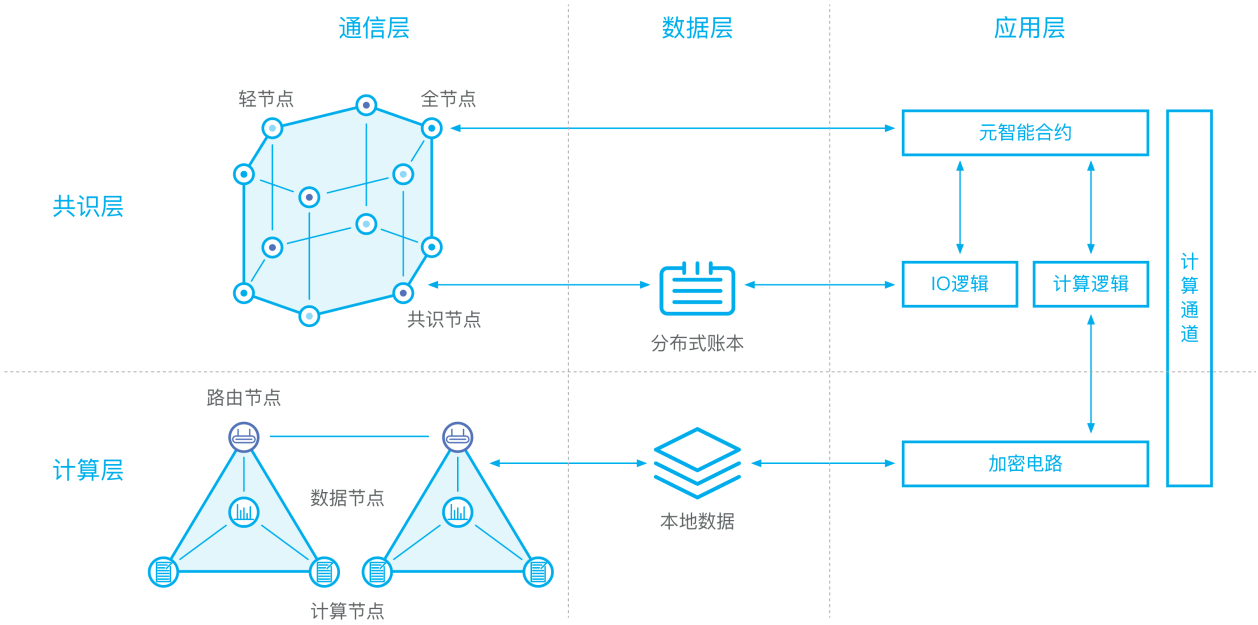

图 3.4　共识与计算解耦

PlatON 中的智能合约包含 IO 逻辑和计算逻辑，IO 逻辑负责读写链上数据，计算逻辑被编译成布尔电路（Boolean Circuit），并分拆为多个并行计算任务，分发到链下的计算节点进行计算。

PlatON 通过随机方式选择匹配的计算节点进行计算，为保证计算的可靠性，同一个子任务会同时分发给多个计算节点，保留一定的计算冗余度。

3.4.2　计算通道

要成为计算节点需要在链上担保一定的数字资产，担保的资产会在计算被计算通道验证正确时返还，欺诈计算发生时担保资产会被扣除。

计算通道是一个系统智能合约，可认为是计算的状态机，负责维护计算的状态，同时也是一个计算法庭，基于可验证计算算法验证计算的正确性，对正确的计算进行清结算和奖励，并对欺诈计算进行惩罚。

3.4.3　多链架构

PlatON 以 Sharding 方式扩展多个应用链，各链业务上相互独立，逻辑上相互平行。多个应用链上的交易并行打包，应用链的区块头在主链上达成共识。

第 4 章

PlatON 技术架构

4.1 PlatON 网络协议

4.1.1 网络协议栈

PlatON 网络的基本实现是全分布式结构化拓扑[1]（Decentralized Structured Topology），完全基于 RELOAD（REsource LOcation And Discovery）基础协议 [RFC6940] 和 Kademlia 协议 [Kademlia]。图4.1为 PlatON 网络整体分层结构。

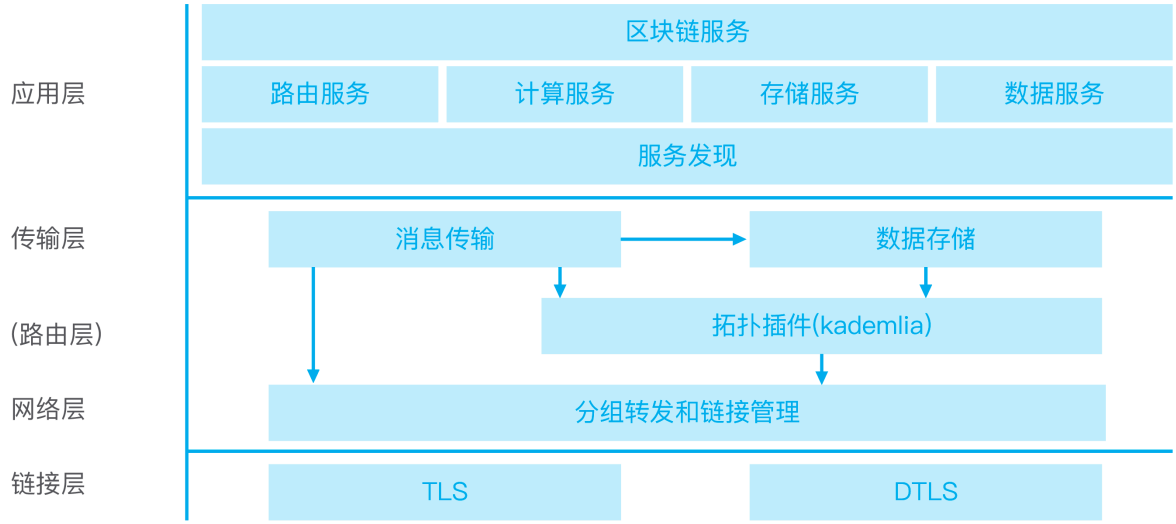

图 4.1　PlatON 网络协议栈（参考自 [RFC6940]）

[1]全分布式结构化拓扑是 P2P 网络的一种拓扑形式，主要采用分布式散列表（Distributed Hash Table, 简写成 DHT）技术来组织网络中的结点。

4.1.1.1 链接层

链接层定位于实现数据的安全传输，提供多种传输协议来防止窃听、篡改、消息伪造；提供安全、可认证的连接；保证消息来源认证和消息数据的完整性。

本层实现安全传输层协议（TLS）和数据包传输层安全性协议（DTLS）。

针对密码算法，PlatON 扩展实现为插件机制，可灵活支持国际标准算法（包括 SHA256、SHA3、ECDSA、RSA、3DES、AES、RSA-OAEP、ECIES 等）。同时将率先支持中国国密算法（SM2、SM3、SM4、SM9 等）。

4.1.1.2 分组转发和连接管理

负责提供分组转发服务来实现存储路由表，同时负责点对点建立连接，包括位于 NAT 设备和防火墙后的节点。RELOAD 使用 ICE 方法 [RFC5245] 实现 NAT 穿越[2]。

4.1.1.3 拓扑插件

RELOAD 是一个 P2P 网络框架，支持扩展不同的拓扑算法来实现全分布式非结构化拓扑或全分布式结构化拓扑网络。

拓扑算法可利用消息传输组件来管理消息的收发，利用存储组件来管理数据的存储。

拓扑算法与分组转发和链接管理层紧密配合，提供多种路由功能来满足不同需求。PlatON 网络采用 Kademlia 算法来实现全分布式结构化拓扑网络。

4.1.1.4 数据存储

负责数据的存储，通过与拓扑插件的配合完成数据的复制、迁移等动作，同时与消息传输组件配合完成数据消息的收发。RELOAD 支持字符串、数组和 dictionary 类型的数据存储。

4.1.1.5 消息传输

负责对应用提供可靠的点对点消息传输服务。PlatON 在 RELOAD 基础上扩展了分区泛洪算法来进行消息的快速全网广播。

4.1.1.6 应用层

利用 RELOAD 底层的通信、存储能力来构建服务发现扩展，以及基于服务发现的 TURN[3]服务、SIP[4]服务、计算服务、数据服务、存储服务、区块链服务等。

---

[2]NAT 穿越（NAT traversal）涉及 TCP/IP 网络中的一个常见问题，即在处于使用了 NAT 设备的私有 TCP/IP 网络中的主机之间建立连接的问题。

[3]TURN(Traversal Using Relay NAT) 是 STUN/RFC5389 的一个拓展，定义了一种中继协议，使得 Symmetric NAT 后面的主机能使用中继服务与对端进行报文传输。

[4]SIP(Session Initiation Protocol) 是由 IETF 制定的多媒体通信协议。SIP 是一个基于文本的应用层控制协议，用于创建、修改和释放一个或多个参与者的会话。这些会话可以是 Internet 多媒体会议、IP 电话或多媒体分发。

以下章节主要描述应用层各服务的网络协议。

4.1.2 服务发现

在 P2P 覆盖网络中，有些节点负责对外部提供服务，有些节点负责向其他节点请求服务，比如中继服务、语音邮件服务、网关定位服务、转码服务等。PlatON 中也需要部分节点提供算力服务、TURN 服务、SIP 服务等。其中服务发现是关键问题所在。

4.1.2.1 ReDiR 树

在 P2P 网络中，最简单的方式是在 DHT[5]中以一个特定的 KEY 保存所有提供某个服务的节点 ID。但使用这种方法，将使存储节点的存储负载过大，而且会导致路由到存储节点的服务查询请求过多，造成消息处理负载过大。

为解决以上问题，PlatON 使用 ReDiR（Recursive Distributed Rendezvous）[RFC7374] 来实现服务发现机制，ReDiR 可以支持数万的服务提供节点及服务查询节点。

ReDiR 使用树状结构实现 P2P 服务发现机制。同时使用 RELOAD 覆盖网络的存储能力保存数据，每一类服务都存储为一棵 ReDiR 树，树节点保存服务提供节点的信息。当某个节点请求查找指定服务的提供者时，对 ReDiR 树做有限次的查找就可以找到与请求节点最匹配的服务提供节点。

ReDiR 树节点使用 RELOAD 的 dictionary 结构存储服务提供节点，每一个 ReDiR 树节点属于 ReDiR 树的某一层（$level$），ReDiR 树的根节点为第 0 层, 根节点的子节点位于第 1 层，第一层的子节点位于第 2 层，以此类推。

ReDiR 树每层容纳的节点数取决于分支因子 b，每层最多容纳 b^{level} 个节点，每个节点用 $(level, j)$ 来唯一标识，其中 $level$ 为节点所在的层数，j 表示该节点为相应层中第 j 个节点。在每一层中，b^{level} 个树节点把第 $level$ 层分为个 b^{level} 个 KEY 空间。

所有服务节点映射存储到相应的 KEY 空间，每个 KEY 空间由一个树节点负责存储，树节点 $(level, j)$ 包含的 KEY 范围为

$$(2^{\texttt{BitsInKEY}} b^{-level}(j + \frac{b'}{b}),\ (2^{\texttt{BitsInKEY}} b^{-level}(j + \frac{b'+1}{b}))$$

其中 $0 \leq b' < b$，树节点 $(level, j)$ 中保存的资源 ID 取值为 $ID = \texttt{hash}(service, level, j)$。

图4.2为分支因子为 2 的 ReDiR 树。

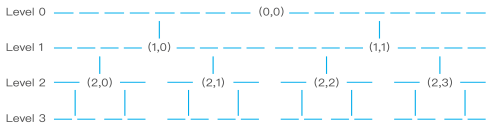

图 4.2　分支因子为 2 的 ReDiR 树（参考自 [RFC7374]）

[5]DHT(Distributed Hash Table，分布式哈希表)，是一种分布式存储方法。在不需要服务器的情况下，每个客户端负责一个小范围的路由，并负责存储一小部分数据，从而实现整个 DHT 网络的寻址和存储。

4.1.2.2 服务发布

在 RELOAD 覆盖网络中，KEY 为 k 的节点 n 发布服务的步骤如下：

- 步骤一：选择一个初始层 $l = l_{start}$，一般为 2。
- 步骤二：节点 n 发送查询请求到负责 KEY 空间 $I(l,k)$ 的树节点，获取该树节点存储的服务节点列表。
- 步骤三：节点 n 发送存储请求将自身信息存储到负责 KEY 空间 $I(l,k)$ 的树节点中。
- 步骤四：检查第一步返回的结果，如果节点 n 的 KEY 值 k 是其中最大或最小的，则将当前层数减 1，重复第 2-3 步，直到节点 n 的 KEY 值不是最大或最小，或者到达根节点为止。

同理，节点 n 从层 $l = l_{start}$ 往下层遍历处理，直到满足以下条件为止：负责 KEY 空间 $I(l,k)$ 的树节点中，节点 n 为唯一一个服务节点。

4.1.2.3 服务更新

注册到 ReDiR 中的服务状态都是动态的，服务节点需要定期重复服务发布流程来更新服务状态。若超时未更新，负责存储的树节点需要将其从存储中删除。

4.1.2.4 服务查找

服务查找过程跟服务发布类似，也是从一个初始层 $l = l_{start}$ 开始，每一步获取到当前 KEY 空间 $I(l,k)$ 中的服务节点列表，按照以下方法处理：

- 步骤一：如果没有返回任何服务节点，则表明 KEY(k) 对应的服务节点存在更大的 KEY 空间，将层数减 1 然后重复查询，如果当前 $level$ 为 0 则查询失败。
- 步骤二：如果在返回的服务节点中，k 不是其中最大或最小的，则表明对应的服务节点一定存在的子空间中，将层数加 1，然后重复查询。
- 步骤三：否则，返回的结果为最接近 KEY(k) 的服务节点，查询成功。

4.1.3 计算服务

如何高效地进行计算节点间通信在 PlatON 中变得十分重要。建立在 RELOAD 协议基础上的计算服务，实现了计算服务的发布、计算服务的发现、计算会话的建立等。

4.1.3.1 发布算力服务

计算节点加入 PlatON 网络提供算力服务，在注册为算力服务提供方之前，需要利用 STUN[6]协议 [RFC5389] 来判断出自己是否在 NAT 设备后面。如果已经在 NAT 设备后面，则需要通过服务发现找到一个 TURN 服务来为自己提供公网的服务能力。计算节点需要把自己的 IP 地址或从 TURN 服务获取到的 Relay 地址注册到 PlatON 网络。

[6]STUN(Session Traversal Utilities for NAT) 是一个轻量级的协议，可以被终端用来发现其公网 IP 和端口，同时可以检测端点间的连接性，也可以作为一种保活（keep-alive）协议来维持 NAT 的绑定。

计算节点在发布算力服务时，使用服务发现扩展协议进行服务发布和更新，将自己发布到 KEY 空间 $I(l, \texttt{power})$ 中，$power$ 为节点提供的算力。

4.1.3.2 发现算力服务

PlatON 计算时，需要根据算力匹配计算节点，使用服务发现扩展协议查找 KEY 空间 $I(l, \texttt{power})$，$power$ 为需要的算力参数。

4.1.3.3 计算任务分发

PlatON 在 RELOAD 基础上封装了计算任务分发协议，把计算任务通过 P2P 通讯分发给提供算力服务的计算节点。为保证计算的可靠性和性能，计算任务分发保持一定的冗余度，即同时分发给多个计算节点。

4.1.3.4 安全多方计算协议

PlatON 在 RELOAD 基础上封装了 GC 和 OT 的协议，以支持安全多方计算。安全多方计算中多个数据方通过 SIP 协议建立计算会话。

4.1.4 区块链服务

区块链服务的节点使用 RELOAD 框架的消息传输组件进行交易数据的转发和区块数据的同步，以及共识过程中的点对点的通讯。

4.1.4.1 区块链节点的加入

多个区块链可以同时运行在 PlatON 网络中，区块链在 PlatON 网络中也作为一种特殊的“服务”存在，节点选择加入指定的区块链时，需要使用服务发布方法将自己发布为指定区块链服务（服务名为指定区块链名称）的提供者。

客户端发起交易时可以使用服务发现方法根据区块链名称查找到指定区块链的节点，并向其发起区块链交易。

4.1.4.2 交易数据的转发

利用 RELOAD 消息传输组件的快速广播能力，区块链交易能够迅速扩散到全网并打包到区块中。

4.1.4.3 区块数据的同步

PlatON 中，每个全节点都保持一份区块链的完整副本，区块经过共识后广播到整个 RELOAD 覆盖网络，各节点接收验证成功后保存到本地。

区块数据使用 RELOAD 消息组件的广播功能进行同步。

4.1.4.4 可验证计算证明共识协议

Giskard 共识算法中，基于计算贡献值加权权益选举出共识节点，该选举过程就是区块链交易。选举出来的多个共识节点通过异步 BFT 协议出块，异步 BFT 协议基于 RELOAD 协议实现。

4.2 PlatON 网络结构

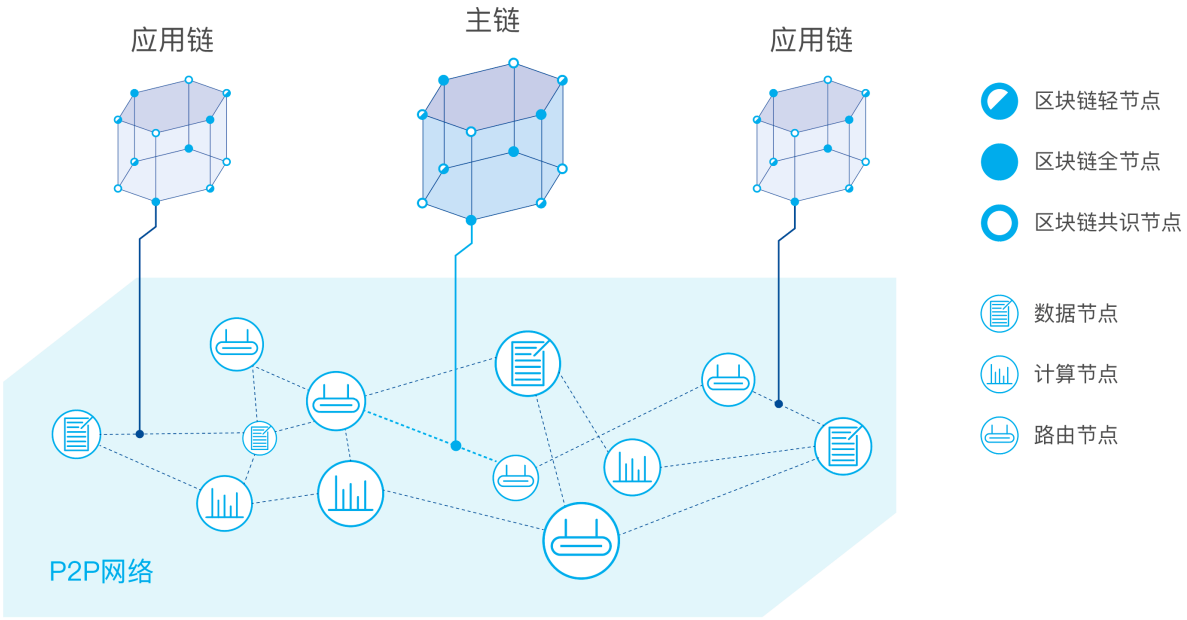

图 4.3 PlatON 网络结构

如图4.3所示，PlatON 网络是一个 RELOAD 覆盖网络，所有节点都需要加入到 RELOAD 覆盖网络；每个节点拥有自己的节点 ID，因此可以在 RELOAD 覆盖网络上根据节点 ID 找到任何节点的路由信息。

在 RELOAD 覆盖网络之上，存在多条逻辑意义上的区块链，每个节点可选择加入某一条或多条区块链，成为该链的节点。在 RELOAD 覆盖网络中，每条区块链就是一类服务；加入一条区块链，也就意味着该节点在 RELOAD 覆盖网络上把自己注册成为相应区块链服务的提供者。

PlatON 网络中的节点还可以独立于区块链之外提供一些特定服务，如 TURN 服务、SIP 服务、计算服务、数据服务、存储服务等。这些服务可以供网络上所有区块链使用。提供服务的节点可以获取不同形式的、可度量的经济激励及社区奖励。

4.2.1 PlatON 节点

按照提供服务类别，PlatON 网络中的节点分为两类：

4.2.1.1 基础服务节点

- **计算节点**
  为 PlatON 网络提供算力服务的节点，负责完成各种计算任务。
- **数据节点**
  为 PlatON 网络提供数据服务的节点，负责为各种计算任务提供数据。
- **路由节点**
  PlatON 网络的节点可以部署在私有网络内，私有网络内的节点可通过路由节点实现 NAT 穿越，路由节点提供 STUN 和 TURN 服务。

4.2.1.2 区块链节点

- **轻节点**
  不保存所有区块的数据，只保存区块头信息以及跟自己相关的数据，依赖全节点进行快速交易验证。轻节点参与交易和区块信息的全网广播。
- **全节点**
  保存了所有区块的数据，可以在本地直接验证交易数据的有效性。全节点参与交易和区块信息的全网广播。
- **共识节点**
  负责执行交易并把交易数据打包成区块。在 Giskard 共识协议中，共识节点基于计算贡献值加权权益选举产生，并通过异步 BFT 协议达成共识。

4.2.2 多链路由机制

在 PlatON 网络中，区块链服务是“一点接入，全网服务”，即 PlatON 用户可以通过网络上任意节点发起到任意区块链的交易，而无需知道目标应用链的节点地址。这意味着区块链路由对于用户来说是透明的。

在 PlatON 网络中，每条区块链就是某一类服务，每个区块链节点加入指定区块链的同时，也将自己注册成为相应应用链的服务提供者。区块链节点使用服务发现协议进行服务发布和更新，将自己发布到相应区块链服务的 KEY 空间 $I(l, NodeID)$ 中，NodeID 为区块链节点 ID。每个区块链服务在整个 PlatON 网络上最终构成一颗 ReDiR 树。

PlatON 用户可以连接到网络上任意节点，通过该节点在指定区块链的 ReDiR 树当中查找 KEY 空间 $I(l, random)$，其中 $random$ 为随机 Hash 值，用于随机查找一个指定区块链节点。查找成功后，就可以直接连接到相应节点向指定区块链发起交易。

4.3 元计算框架 Monad

计算本质上由算法、数据和算力构成。

PlatON 元计算框架的目的就是有效整合全球范围内异构的算法资源、数据资源、计算资源，从而深刻而广泛地促进数据交易和算力交易。

PlatON 采用的元计算框架在使用并行计算和专用计算硬件提高计算性能的同时，也集成了多种密码学算法保证计算的可验证和数据隐私。

PlatON 元计算框架集成了安全多方计算、可验证计算、同态加密、零知识证明等密码学算法。

4.3.1 元计算定义

Monad 是一种区块链扩容计算的实现。通过将计算扩展到链下，使得算力可以线性扩容；使用并行计算和专用计算硬件提高计算性能的同时，集成多种密码学算法来保证计算的可验证性和数据隐私。

4.3.2 元计算参与方

在元计算框架中，计算参与方按能力类型分为计算发起方、算法提供方、数据提供方、算力提供方、计算协调方。

4.3.2.1 计算发起方 U

一般指外部用户，其通过客户端发起元智能合约的调用，从而触发计算。

4.3.2.2 算法提供方 A

特指元智能合约的发布者。算法包含在元智能合约中，其定义了计算逻辑和输入输出数据格式，计算逻辑被编译成布尔电路（Boolean Circuit）。算法随元智能合约一起发布到 PlatON 平台。

4.3.2.3 数据提供方 D

在 PlatON 平台中，数据提供方也称为数据节点，数据保存在数据节点的本地数据库中。数据提供方根据算法定义的输入数据格式，提供相应数据用于计算。共识节点是一类特殊的数据提供方，共识节点实际上是链上数据的提供方。

4.3.2.4 算力提供方 P

接收并执行计算任务（包含算法和数据）。在 PlatON 网络中也称为计算节点。

4.3.2.5 计算协调方 C

计算协调方负责获取数据，并将数据和计算逻辑一起构成计算任务分发给算力提供方进行计算。计算协调方一般而言也是数据提供方。

4.3.3 元计算任务

4.3.3.1 计算数据源

元计算任务的数据可分为单源数据和多源数据。

- **单源数据**
  数据来源于单个数据提供方，该数据提供方如果是共识节点，则数据来源于链上。
- **多源数据**
  数据来源于多个数据提供方，包括链上和链下的多个数据提供方。

4.3.3.2 计算分类

元计算支持多种形式的计算任务。可分为可验证代理计算和隐私代理计算。

- **可验证代理计算**
  计算过程中不需要保证数据的隐私，当算力提供方完成计算后，在返回计算结果的同时也返回正确执行的证明，由计算发起方进行验证。
  对多源数据而言，各数据提供方各自进行可验证代理计算。
- **隐私代理计算**
  计算过程中需要保证数据的隐私。单源数据和多源数据都适合进行隐私代理计算，但是使用的密码算法不一样，后面章节将会分开阐述。

4.3.4 计算通道

PlatON 的计算通道（Computing Channels）实现为一个特殊合约，该合约通过可验证计算和时间承诺保证计算的有效性。

计算通道预扣交易发起方一定量的 Energon 作为计算任务的质押，任务结束且完成结算之后，再将剩余的 Energon 返回给交易发起方。

计算结束时，每一个参与方都可以往计算通道合约中发送一个交易，从而关闭这个通道并启动一个结算过程。但计算通道不会马上进行结算，会先启动一个时间区间，在该区间内任何参与方均可对于计算过程提交异议申诉。

因此区块链就可以被视为某种“密码学法庭”，作为 PlatON 网络中一种成本较高的仲裁手段。这是用于提供交易安全性的最后手段，也因此极大地降低了造假和欺诈的动机。当然，经过长期验证的有效工作量证明和交易结果证明已经可以很好地保证计算的有效性，这种仲裁手段仅作为交易安全的强有力补充和最终保障。

4.3.5 计算任务执行

4.3.5.1 计算资源分配

当计算节点加入网络时，将自己发布为计算服务。发布服务时，该节点会自动检测和评估自己的计算能力并发布为**服务能力参数**。

当**计算协调方**试图发起计算任务的分发时，首先需要从网络上查找能力匹配的计算节点。使用**服务发现协议**在网络上随机查找符合算力需求的计算节点，并根据计算贡献值排

序，选择计算贡献值高的计算节点。为保证公平性，鼓励更多的节点提供算力服务，还将随机选择一部分新加入的计算节点。

4.3.5.2 可验证代理计算

算法提供方在发布计算逻辑时指定“可验证”为计算需求。合约执行时，计算协调方配置 VC 算法的相应参数，并将合约中的布尔电路拆分成多个子电路。

计算协调方将 VC 算法参数、子电路以及输入数据整合成多个子任务，并分发给多个计算节点。为保证计算的可靠性，同一个子任务会同时分发给多个计算节点，从而保留一定的计算冗余度来做相互校验。

计算节点在进行子任务计算的同时，利用 VC 算法生成正确执行的证明，并返回给计算协调方。证明验证通过后，计算协调方按照算力提供方计算的门电路个数，累加加权计算贡献值之后按照一定比例分配奖励。可验证代理计算流程如图4.4。

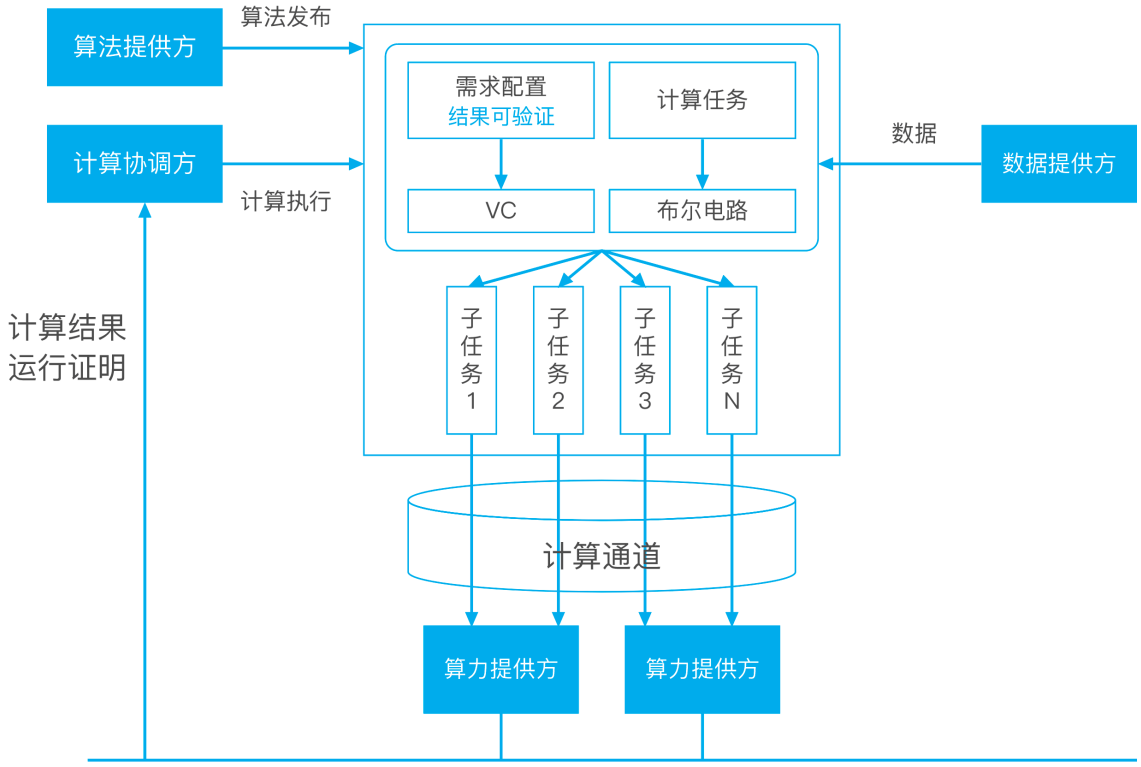

图 4.4 可验证代理计算流程

4.3.5.3 单源数据隐私代理计算

单数据源可在不泄露原始数据的前提下利用全网的计算资源。当合约执行时，**计算协调方**配置同态算法以及 VC 算法的相应参数，并将合约中的布尔电路拆分成多个子电路。

计算协调方将数据进行同态加密后结合 VC 算法参数把计算任务拆解为多个子任务，并分发给多个计算节点。为保证计算的可靠性，同一个子任务会同时分发给多个计算节点，保留一定的计算冗余度。

计算节点根据子电路进行密文的同态计算，同时采用 VC 算法证明其执行的正确性。计算节点把加密的计算结果以及计算证明返回给计算协调方。计算协调方首先在验证证明的有效性后，对密文进行解密获取结果。计算协调方按照算力提供方计算的门电路个数，累加计算贡献值，并按一定比例分配奖励。图4.5为单源数据隐私计算流程。

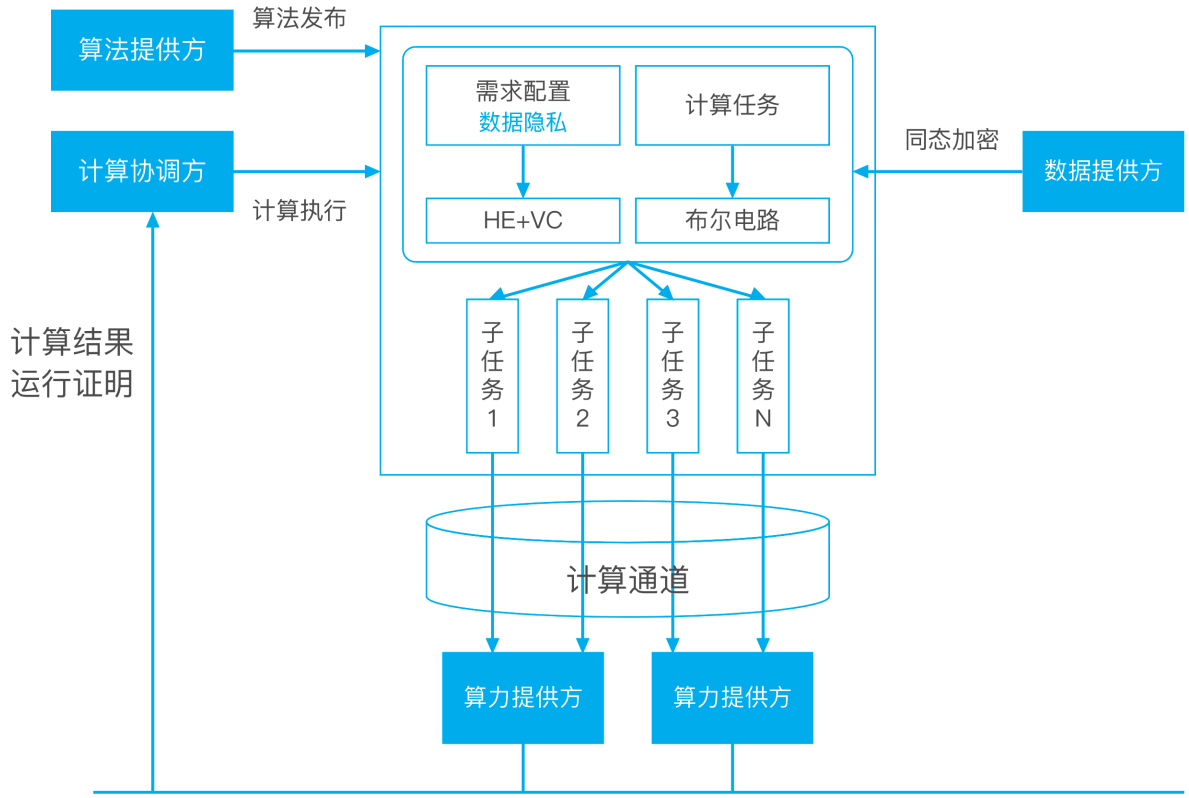

图 4.5　单源数据隐私计算流程

4.3.5.4　多源数据隐私代理计算

多个数据源可协同发起计算任务得到计算结果，并且保证各方对数据的所有权。多源数据计算的隐私保护采用 MPC 算法。

安全多方计算采用经典的 Yao[7]的加密电路[8]（Garbled Circuit）方法。简单而言，加密电路方法有一个生成方和执行方。

生成方按照电路将电路中的每一条线选取一个随机数作为标签（label），并按照电路的逻辑对每一个门用输入的标签加密输出的标签。最后将普通的布尔电路转化为加密电路。

[7]姚期智（Andrew Chi-Chih Yao）先生, 世界著名计算机学家，美国科学院院士，美国科学与艺术学院院士，中国科学院外籍院士。由于对计算理论包括伪随机数生成、密码学与通信复杂度的突出贡献，2000 年获得图灵奖。

[8]加密电路（Garbled Circuit）最早由姚期智先生提出的对电路进行加密的方法，是安全多方计算中最常用的工具之一。

执行方与生成方之间运行 OT 协议获取与双方输入相关的标签，解密电路最终获取计算结果。

合约执行时，共识节点作为 GC 生成方发布计算任务，配置 GC 生成算法和 VC 算法参数。共识节点随机选取标签（标签的选法与具体的 GC 算法相关），将计算任务按子电路及标签分拆成多个子任务并分发给多个计算节点计算，最终得到多个加密子电路。计算节点在进行子任务计算时，同样需要用 VC 算法生成正确执行的证明，并返回给共识节点验证。

同时，数据提供方作为 GC 执行方与 GC 生成方之间运行 OT 协议，获取和其输入相关的标签。这个过程可以和 GC 生成方发布计算任务并行进行。

GC 执行方在获得标签和加密电路后，可以再发布计算任务对电路进行解密。其发布方式与 GC 生成方发布的方式类似，只是为其计算需求配置的为 GC 解密算法。

通过将消耗资源的电路加密和解密过程以计算任务的形式分发给计算节点，利用激励机制引导各种算力来参与进行并行计算。将可以充分利用闲置计算资源，极大的促进数据的安全流动和隐私保护。多源数据隐私计算流程如图4.6。

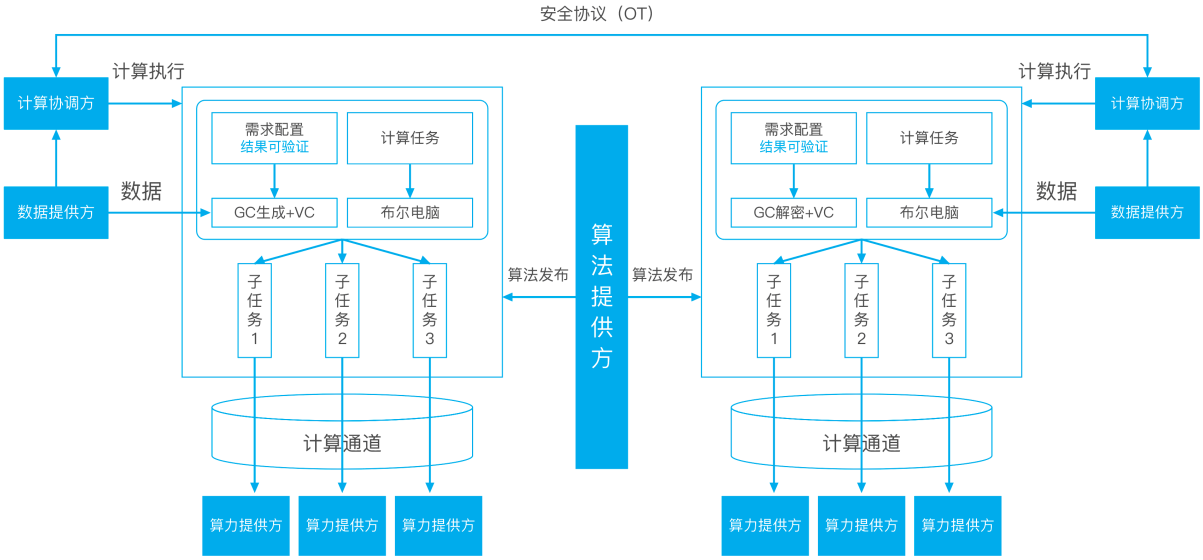

图 4.6　多源数据隐私计算流程

4.3.6　专用计算硬件

在 PlatON 中，每个元智能合约的计算逻辑都被编译为相应的布尔电路，构成布尔电路的基本门电路天然地和硬件架构匹配，通过硬件平台来执行布尔电路将可以极大地提高运算效率，并降低功耗及成本。

PlatON 将会分为以下几个阶段来推进专用计算硬件的研发与部署：

- **第一阶段**：设计适用的硬件 IP，并利用 FPGA 实现相应的硬件设备；
- **第二阶段**：将 IP 授权给 ASIC/ASSP[9]/SoC 厂商，协助这些厂商集成在其设计中，支持 PlatON 的计算框架加速；

[9]ASSP(Application Specific Standard Parts，专用标准产品) 是为在特殊应用中使用而设计的集成电路。

- **第三阶段：**实现 ASIC 芯片，为生态合作伙伴提供芯片级的解决方案。

PlatON 将会秉承开放态度与生态体系的各类合作伙伴充分合作，以推动下一代计算架构的全面部署。我们坚定地相信只有通过软硬件的协同进化才能彻底地提升网络性能，以此践行我们对于下一代计算架构的设想。

4.4 可验证计算证明共识 Giskard

为避免算力浪费和提高共识效率，Giskard 共识不采用 PoW 方式，PlatON 会以选举加随机的方式选取部分节点参与共识。

为选取真正积极参与和诚实计算的节点，Giskard 的选举以计算贡献值为依据。计算贡献值可衡量节点的贡献度和诚实度。

4.4.1 计算贡献值

PlatON 采用多维度、多属性的账户体系。账户体系中包含资金余额、计算贡献值等属性。这样的账户体系可更准确地刻画用户在整个网络中的行为画像。

计算贡献值是用户提供的经过验证的有效算力。PlatON 中所有计算都展开为电路化，计算的消耗与门电路的个数成正比，并且每个门的消耗权重也各不相同。PlatON 为每个门电路定义了计算贡献点，节点的计算贡献值为所有计算贡献点的累计值。

4.4.2 可验证计算证明共识

“劳心者治人，劳力者治于人”。这一治理架构在全数字化世界将会面临全新的解构和解读。

计算是一切事务的基本过程和本质。劳心者与劳力者貌似分处不同阶层，但本质上都参与或者发起了计算过程。

在 PlatON 网络中，我们相信劳动创造价值。劳心者的一切权力源于劳力者，也应当从劳力者中产生，其所产生的所有决策也应当且需要满足劳力者的自然权利。当然劳动创造价值的方式是多样化的，作为一个有包容性的社区，应该兼容包括计算贡献值维度在内的多种衡量标准，来做综合加权评估。这一理念将通过恰当的算法机制予以保障。

在 PlatON 网络中，一部分共识节点通过选举产生，一部分共识节点采用可验证随机函数[10]（VRF）随机选取。选举采用持续实时投票方式，选票根据包括计算贡献值在内的多维度因素进行综合加权评估，每隔一个时间周期根据投票情况重新确定下一轮参与共识的节点。

共识节点采用异步 BFT 算法出块并达成共识。恶意节点将被取消其作为共识节点的资格，并给与一定的计算贡献值扣除及经济惩罚。共识节点在执行打包交易时，将交易中的计算逻辑拆分到多个计算节点并行计算，每个计算节点在反馈计算结果的同时返回正确执行的证明。共识节点将结果与证明打包到区块中，其他节点只需验证证明确定区块的合法性，可以大大减少区块验证时间，提高交易性能。

[10]可验证随机函数（VRF）的概念由 Micali，Rabin 和 Vadhan 提出。它是一种伪随机函数，可以提供其输出正确性的公开可验证的证明。

候选节点的选举、投票和出块机制不仅是一个技术问题，更多的是社区理念和治理模式选择问题。在本技术白皮书中不做过多探讨，将会根据社区反馈意见在适当时机发布的生态治理方案中提出。

4.5 元智能合约 Sophia

PlatON 所采用的智能合约体系完全不同于传统智能合约。PlatON 将致力于服务计算世界，创造性地提出面向计算的“元智能合约”。

4.5.1 元智能合约分类

PlatON 本质上是一个去中心化拓扑结构下的 Serverless 架构的分布式服务平台。元智能合约就是部署于其上的 FaaS[11]（Functions as a Service）应用。

用户只需要提交元智能合约代码到 PlatON 平台，就可以对外发布服务，且只需要为执行代码过程中消耗的资源付费。

PlatON 将元智能合约分为以下三种类型：

4.5.1.1 状态合约

状态合约类似于传统的智能合约。

这类智能合约需要在链上保存状态，状态合约执行时输入的数据来自链上分布式账本，每次合约执行会导致合约的状态变更，所有变更均会被记录在分布式账本中。

如图4.7所示，合约的计算拆分成多个子任务分发给多个计算节点，合约开发者可以选择隐私计算的方式，以保证数据不透露给计算节点。

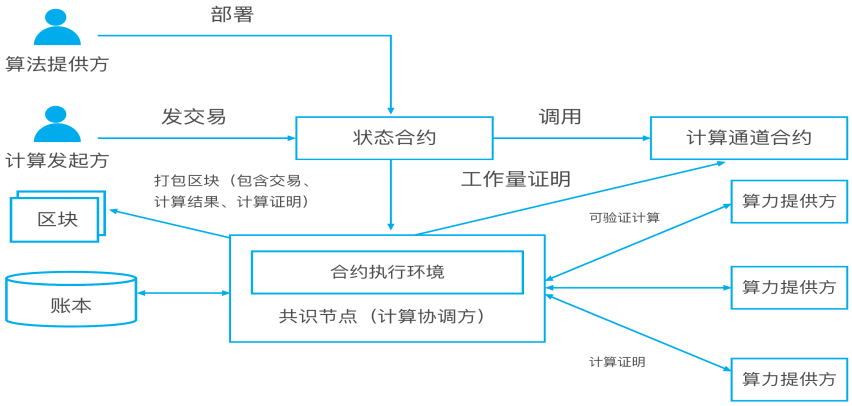

图 4.7 状态合约

[11]FaaS(Functions as a Service) 是一种无服务架构 (Serverless Architecture)，FaaS 将函数转换为无状态服务，同时管理服务的生命周期，FaaS 的每个函数都拥有快速启动和短暂生命周期的特性，在运行的时候才消费资源。

4.5.1.2 无状态合约

无状态合约在链上不保存任何状态。

如图4.8所示，无状态合约执行时输入的数据来自链下数据提供方的本地数据库，可以是单个数据提供方，亦可是多个数据提供方。

单数据提供方的计算过程跟有状态合约相同;如果是多个数据提供方,则多方使用 MPC 算法进行协同计算，保证各方对数据的所有权。将实际的计算拆分成多个子任务分发给多个计算节点，合约开发者可以选择隐私计算的方式，以保证数据不透露给计算节点。

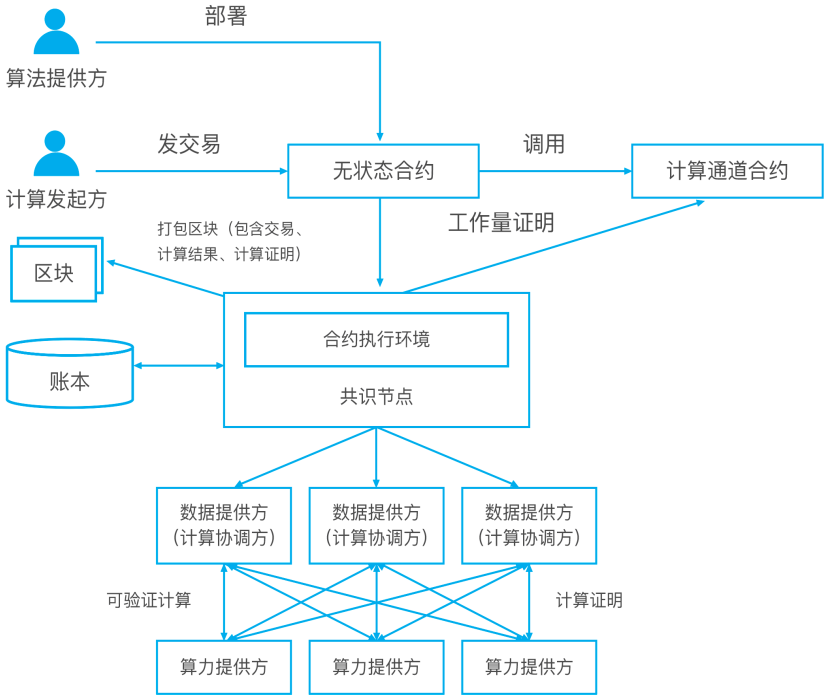

图 4.8　无状态合约

4.5.1.3 混合合约

PlatON 也允许智能合约在链上保存状态，又存在有链下数据参与计算，这类合约称为混合合约。

混合合约本质上是一种多源数据计算，共识节点作为链上状态数据的数据提供方参与计算。混合合约典型的使用场景是把链下的数据计算结果保存到链上，并参与到下一次的计算中。

4.5.2 元智能合约虚拟机

为适配并行计算和专用计算硬件，加速计算性能，PlatON 把元智能合约编译成布尔电路来执行。元智能合约使用 Java 语言开发，PlatON 提供编译器将元智能合约中的计算逻辑编译成布尔电路。

除了提供能直接运行元智能合约的专用计算硬件，PlatON 也提供元智能合约虚拟机，可在各种软硬件环境下运行元智能合约，也供其他区块链项目集成使用。

第 5 章

能量块 Energon

PlatON 由一条主链和多个应用链构成。各链业务上相互独立，逻辑上相互平行。主链是 PlatON 网络的初始链，应用链是为解决特定行业问题而衍生出来的垂直链。

5.1 能量块 Energon

PlatON 是一个基于服务的计算架构，除了提供计算、数据、存储、网络等基础服务外，应用开发者也可在 PlatON 上发布自己的应用服务。PlatON 上每个应用的运行都要消耗一定的资源（包括算力、带宽、存储、数据等）。为实现资源的公平合理使用，避免资源的滥用，PlatON 通过一系列算法实现资源的合理调度和有效性验证，并使用 Energon 来度量资源的使用。Energon 也是驱动 PlatON 这个“计算工厂”运转的能量。各应用链也可以创建自己的应用 Energon。在 PlatON 网络中，用户只需要一个统一账户就可以管理和使用自己的 Energon。不同链的 Energon 可以自由跨链转移。

5.2 能量块交换合约

PlatON 网络中 Energon 的跨链转移是通过名为能量块交换合约（Energon Exchange Contract，简称 EEC）的特殊合约来进行的，EEC 合约内置在主链和各应用链中。EEC 智能合约中使用双向锚定 [Two-way Peg] 和原子交换 [Atomic Swaps] 技术实现 Energon 的跨链转移。

第 6 章

技术路线图

PlatON 项目的技术路线如图6.1所示：

图 6.1　PlatON 技术路线图
【里程碑实现日期根据实际情况相应调整】

- **贝莱世界（Baleyworld）：**
  实现完整的 RELOAD 覆盖网络和区块链服务，支持服务发现、元智能合约以及 VC 算法。
- **川陀（Trantor）：**
  实现多方安全计算、优化的 VC 算法以及 Giskard 共识。
- **端点（Terminus）：**
  实现链下并行计算。
- **盖娅（Gaea）：**
  实现软硬件一体化，发布专用计算硬件。

第 7 章

社群的进化

7.1 技术的进化

PlatON 网络是一个处在开发初期、尚不足够完备的系统。目前给出的只是 PlatON 作为下一代计算架构的基本描述。必须指出的是，PlatON 创造的复杂网络面临巨大的技术挑战，无论是分布式架构、密码学算法、博弈论机制的设计、硬件实现和网络建设都存在诸多问题，有待于学术界的理论突破和工程上的点滴探索。有些问题甚至是全人类共同面临的智力挑战。我们将依托全球社区的力量逐个解决、持续进化，并在未来的路线图中不断更新迭代。

7.2 组织的进化

组织的进化某种程度上就是治理模式的进化。我们相信改良的力量，而不妄谈颠覆。人类社会在不同的组织结构和治理模式中试错、选择，以此循环往复求得最终的进步。

区块链开源社区在过去几年中也遭遇了多次类似的问题，社区的分裂、分叉、黑客攻击、合约漏洞、合规性挑战等等，但仍然保持了鲜活旺盛的生命力。

作为一个全球化的自组织形态的社区，PlatON 在治理模式的设计和实践过程当中也会面临同样多的挑战，但会持续秉承共治、共享、共识的基本理念来解决所面临的治理挑战和异常。

PlatON 的参与方也会从现有提供智能合约的开发者社区、提供算法和理论的学术社区、提供算力的计算社群、提供数据的数据社群和需求方，递次演进至更多参与方和参与者。其间利益必然有所不同，产生的矛盾分歧也必然不都是技术或者算法可以解决的。我们将会根据社区各个群体的反馈，逐步梳理和发布社区治理方案以适应未来。

PlatON 作为一个复杂网络，不会刻意偏袒任何一方，只会一如既往地鼓励和支持更多方、更多机构、更多利益群体、更多个人参与这一网络。越是如此“复杂”，网络就越是强大和健壮。

我们坚信在数据主权日益彰显的时代：**“部分不知道整体、整体不知道部分”**。这既是密码学理论对隐私保护的实现，也是共识机制的价值之所在。

PlatON 组织的进化，其逻辑基础源于共识，源于“同意的计算”。

7.3 网络的进化

PlatON 依托于全网共识、依赖于全球算力、数据、算法的共享、建构在社区共治的基础之上。其中网络基础设施是举足轻重的一部分。

我们将从现有互联网出发，始终跟随全球移动互联网、天基互联网建设运营的步伐，支持、适配乃至于发起各种不同类型的网络基础设施和计算基础设施，最终目标是基于空、天、地一体的网络基础设施来推进 PlatON 的进步。

PlatON 的网络进化过程，将会极大拓宽社区的视野，持续提升社区达成共识的能力。具体工作计划将会以网络建设白皮书的形式在适当时机发布，并定期更新。

脚踏实地，尔后，仰望星空。

术语表

英文	中文	缩写	释义
Algorithm Provider	算法提供者	-	特指元智能合约的发布者。算法包含在元智能合约中，其定义了计算逻辑和输入输出数据格式。
Application Specific Integrated Circuit	专用集成电路	ASIC	专用集成电路是针对整机或系统的需要，专门为之设计制造的集成电路。
Application Specific Standard Parts	专用标准产品	ASSP	ASSP(专用标准产品) 是为在特殊应用中使用而设计的集成电路。
Atomic Swaps	原子交换	-	原子交换（Atomic Swaps）是 T. Nolan 在比特币开发社区提出来的一种代币点对点交易方案。PlatON 中，原子交换用于实现 Energon 在多链间的安全交换。
Block Producer	共识节点	-	负责执行交易并把交易数据打包成区块。在 Giskard 共识协议中，共识节点基于计算贡献值加权权益选举产生，并通过异步 BFT 协议达成共识。
Byzantine Fault Tolerance	拜占庭容错	BFT	拜占庭将军问题首次由 Leslie Lamport，Robert Shostak 和 Marshall Pease 在 1982 年提出。具备拜占庭容错能力的分布式网络能减轻恶意节点对网络的影响并在诚实节点间达成正确共识。目前有几类 BFT 协议可以提高系统拜占庭容错能力，Miguel Castro 和 Barbara Liskov 提出的实用拜占庭容错（PBFT）是这些协议之一。
Circuit	电路	-	一种通用的计算表现形式，由不同类型的门（gate）组成。由逻辑门构成则成为布尔电路（Boolean circuit），由算术门构成则叫算术电路（Arithmetic circuit）。

Computing Requester	计算发起方	-	一般指外部客户端，其通过客户端发起元智能合约的调用，从而触发计算。
Computing Channels	计算通道	-	计算通道作为计算的控制验证层和结算层用于保证计算的有效性和算力交易的安全性。
Computing Node	计算节点	-	为 PlatON 网络提供算力服务的节点，负责完成各种计算任务。
Computation Task	计算任务	-	PlatON 中，计算逻辑编译成布尔电路，并拆分成子电路进行并行计算，每个子电路及其输入打包成一个计算任务。
Computing Collaborator	计算协调方	-	计算协调方负责获取数据，并将数据和计算逻辑一起构成计算任务分发给算力提供方进行计算。
Computing Power Provider	算力提供方	-	对外发布算力服务，接收并执行计算任务。
Data Node	数据节点	-	为 PlatON 网络提供数据服务的节点，负责为各种计算任务提供数据。
Data Provider	数据提供方	-	数据提供方根据算法定义的输入数据格式，提供相应数据用于计算。
Decentralized Application	去中心化应用	Dapp	Dapp 是运行在底层区块链平台上的分布式应用。Dapp 是基于智能合约的应用，由智能合约和前端 APP 构成，智能合约运行在去中心化的区块链节点上。
Decentralized Structured Topology	全分布式结构拓扑	-	全分布式结构化拓扑是 P2P 网络的一种拓扑形式，主要采用分布式散列表（Distributed Hash Table, 简写成 DHT）技术来组织网络中的结点。
Energon	能量块	-	Energon 是 PlatON 资源使用的度量衡，也是驱动 PlatON 这个“计算工厂”运转的能量。PlatON 上每个应用链可独立创建自己的 Energon。

Field-Programmable Gate Array	现场可编程门阵列	FPGA	FPGA 是指一切通过软件手段更改、配置器件内部连接结构和逻辑单元，完成既定设计功能的数字集成电路。FPGA 是作为专用集成电路（ASIC）领域中的一种半定制电路而出现的，既解决了定制电路的不足，又克服了原有可编程器件门电路数有限的缺点。
Full Node	全节点	-	保存了所有区块的数据，可以在本地直接验证交易数据的有效性。
Functions as a Service	功能即服务	FaaS	FaaS 是一种无服务架构 (Serverless Architecture)，FaaS 将函数转换为无状态服务，同时管理服务的生命周期，FaaS 的每个函数都拥有快速启动和短暂生命周期的特性，在运行的时候才消费资源。
(Fully) Homomorphic Encryption	（全）同态加密	(F)HE	在密文上进行计算，既能保证隐私又能提供可操作性。全同态是指支持所有操作的计算。
Garbled Circuit	加密电路	GC	最早由姚期智先生提出的对电路进行加密的方法，是安全多方计算中最常用的工具之一。
Interactive Connectivity Establishment	-	ICE	ICE 是一个用于在 offer/answer 模式下的 NAT 传输协议, 主要用于 UDP 下多媒体会话的建立, 其使用了 STUN 协议以及 TURN 协议, 同时也能被其他实现了 offer/answer 模型的的其他程序所使用, 比如 SIP。
Kademlia	-	-	Kademlia 是由 Petar Maymounkov 与 David Mazières 所设计的 P2P 重叠网络传输协议，以构建分布式的 P2P 电脑网络。是一种基于异或运算的 P2P 信息系统。它制定了网络的结构及规范了节点间通讯和交换资讯的方式。

Light Node	轻节点	-	不保存所有区块的数据，只保存区块头信息以及跟自己相关的数据，依赖全节点进行快速交易验证。
Malicious model	恶意模型	-	一种安全模型，指攻击者可不按照协议规定执行，主动篡改协议、试探诚实参与方，以期获取更多的信息。
Meta Computing Framework	元计算框架	-	元计算框架有效整合全球范围内异构的算法资源、数据资源、计算资源，从而深刻而广泛的促进数据交易和算力交易。元计算框架使用并行计算和专用计算硬件提高计算性能的同时，也集成了多种密码学算法保证计算的可验证和数据隐私。
Meta Smart Contract	元智能合约	-	PlatON 中的智能合约，不同于传统智能合约，元智能合约同时支持对链上链下数据的访问。
Oblivious Transfer	不经意传输	OT	一种安全的传输协议，允许两方对按照输入比特安全选取标签，是安全多方计算中常用的工具之一。
Proof-of-Verifiable-Computation Consensus	可验证计算证明共识	Giskard	PlatON 的共识机制，Giskard 基于计算贡献值选举出共识节点，共识节点使用异步 BFT 协议出块。Giskard 有效解决算力浪费和算力集中的问题。
Proof-of-Work Consensus	工作量证明	PoW	PoW 机制最早应用于 Adam Back 1996 年提出的 Hashcash 中，而后被中本聪改造为以“挖矿”形式实现区块链一致性的共识机制。PoW 中，矿工通过计算符合要求的区块哈希值竞争生成新区块的资格，同时获得相应的币作为奖励。而这整个过程中，矿工贡献的算力就是上面所说的“工作量”。

Recursive Distributed Rendezvous	-	ReDiR	ReDiR 在 RELOAD 基础上定义了一种服务发现机制，已经成为 RFC7374 标准。
REsource LOcation And Discovery	-	RELOAD	RELOAD 协议由 IETF P2PSIP(Peer-to-Peer Session Initiation Protocol) 工作组提出的一个 P2P 网络协议框架提案，已经成为 RFC6940 标准。RELOAD 协议定义了统一的叠加网对等体和客户端协议，实现抽象的存储和消息路由服务。
Routing Node	路由节点	-	PlatON 网络的节点可以部署在私有网络内，私有网络内的节点可通过路由节点实现 NAT 穿越，路由节点提供 STUN 和 TURN 服务。
Secure Multi-Party Computation	安全多方计算	MPC	无可信第三方场景下多个参与方协同计算，获取计算结果，并不泄露各自输入信息。
Session Initiation Protocol	-	SIP	SIP 是由 IETF 制定的多媒体通信协议。SIP 是一个基于文本的应用层控制协议，用于创建、修改和释放一个或多个参与者的会话。这些会话可以是 Internet 多媒体会议、IP 电话或多媒体分发。
Session Traversal Utilities for NAT	-	STUN	STUN 是一个轻量级的协议，可以被终端用来发现其公网 IP 和端口，同时可以检测端点间的连接性，也可以作为一种保活（keep-alive）协议来维持 NAT 的绑定。
System-on-a-Chip	系统级芯片	SoC	SoC 称为系统级芯片，也有称片上系统, 意指它是一个产品，是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容。

Traversal Using Relay NAT	-	TURN	TURN 是 STUN/RFC5389 的一个拓展，定义了一种中继协议，使得 Symmetric NAT 后面的主机能使用中继服务与对端进行报文传输。
Two-way Peg	双向锚定	-	双向锚定 (Two-way Peg) 是 2014 年 Adam Back 等人提出来的一种侧链技术。PlatON 中，双向锚定用于实现 Energon 在多链间的安全转换。
value of computing contribution	计算贡献值	-	计算贡献值是用户提供的经过验证的有效算力。
Verifiable Computation	可验证计算	VC	可有效验证结果数据是否按照原始数据依照指定逻辑计算而来。
Verifiable Random Function	可验证随机函数	VRF	可验证随机函数（VRF）的概念由 Micali，Rabin 和 Vadhan 提出。它是一种伪随机函数，可以提供其输出正确性的公开可验证的证明。
Zero-Knowledge Proof	零知识证明	ZKP	证明者让验证者确信某个事实的正确性，并不泄露其他任何信息（零知识）。

参考文献

【Trustless 计算模型】

[**The Verifier’s Dilemma**] L. Luu, J. Teutsch, R. Kulkarni, P. Saxena. “Demystifying Incentives in the Consensus Computer”. CCS, 2015

[**The Scalability Trilemma**] https://github.com/ethereum/wiki/wiki/Sharding-FAQs this-sounds-like-theres-some-kind-of-scalability-trilemma-at-play-what-is-this-trilemma-and-can-we-break-through-it

[**Foreshadow**] https://foreshadowattack.eu/

【PlatON 网络协议】

[**RFC6940**] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “REsource LOcation And Discovery (RELOAD) Base Protocol”, RFC 6940, January 2014, http://www.rfc-editor.org/info/rfc6940.

[**RFC5245**] J. Rosenberg , “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols”, RFC 5245, April 2010, http://www.rfc-editor.org/info/rfc5254.

[**RFC7374**] J. Maenpaa, and G. Camarillo, Ericsson, “Service Discovery Usage for REsource LOcation And Discovery (RELOAD)”, RFC 7374, October 2014, http://www.rfc-editor.org/info/rfc7374.

[**RFC5766**] R. Mahy, P. Matthews, and J. Rosenberg, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)”, RFC 5766, April 2010, http://www.rfc-editor.org/info/rfc5766.

[**RFC7890**] D. Bryan, P. Matthews, E. Shim, D. Willis, and S.Dawkins, “Concepts and Terminology for Peer-to-Peer SIP (P2PSIP)”, RFC 7890, June 2016, http://www.rfc-editor.org/info/rfc7890.

[**Kademlia**] P. Petar, Maymounkov, David Mazieres. Kademlia: A peer-to -peer information system based on the XOR metric[DB/OL]. www.cs.rice.edu/conferences IPTPS02/109.pdf.

【能量块 Energon】

[**Atomic Swaps**] T. Nolan, Re: Alt chains and atomic transfers, https://bitcointalk.org/index.php topic=193281.msg2224949 msg2224949, 2013.

[**Two-way Peg**] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, Enabling Blockchain Innovations with Pegged Sidechains, <https://blockstream.com/sidechains.pdf>, 2014.

【同态加密】

[1] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, 1978.

[2] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. STOC, 2009.

【零知识证明】

[1] S. Goldwasser, S. Micali, C. Rackoff , "The knowledge complexity of interactive proof systems" , SIAM Journal on Computing, 1989.

[2] B. Manuel, F. Paul, M. Silvio "Non-Interactive Zero-Knowledge and Its Applications". STOC, 1988.

【安全多方计算】

[1] A. C. Yao, Protocols for Secure Computations (Extended Abstract). FOCS, 1982.

[2] A. C. Yao, How to Generate and Exchange Secrets (Extended Abstract). FOCS, 1986.

[3] O. Goldreich, S. Micali, A. Wigderson:How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. STOC, 1987.

【可验证计算】

[1] S. Micali, "Computationally Sound Proofs". SIAM Journal on Computing, 2000.

[2] B. László, F. Lance, L. A. Leonid, S. Mario, "Checking Computations in Polylogarithmic Time". STOC, 1991.

[3] S. Goldwasser, Y. T. Kalai, G. N. Rothblum. "Delegating Computation: Interactive Proofs for Muggles".STOC, 2008.

[4] G. Rosario, G. Craig, P. Bryan. "Non-Interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers". CRYPTO, 2010