Taproot Consensus 黄皮书

第一章：摘要

比特币网络的非图灵完备性质限制了其直接实现类似以太坊 Rollup 的 Layer2 扩展方案。比特币网络的脚本合约层只能进行简单的转账操作，无法支持更复杂的智能合约功能。因此，单纯从比特币脚本层面来构建 Layer2 扩展方案是不可行的。

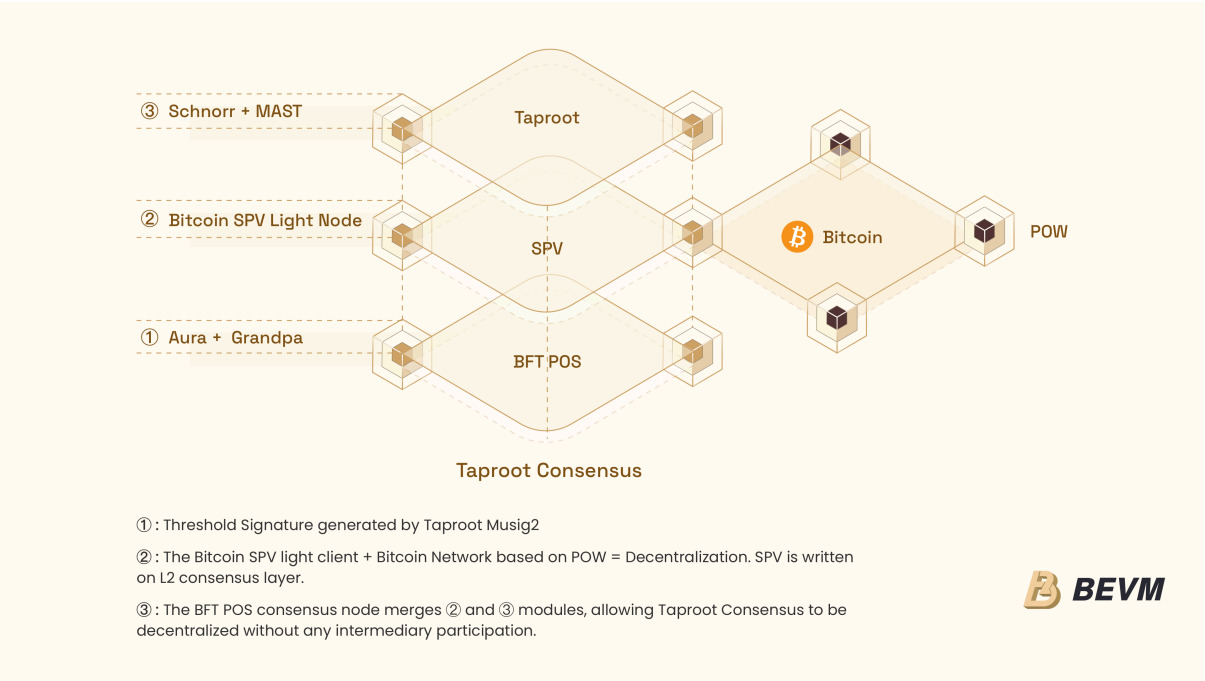
现有的比特币 Layer2 解决方案，如闪电网络，都依赖于外部分布式节点来维护状态通道的账本。这种妥协是闪电网络未能大规模应用的一个根本原因，因为缺乏拜占庭容错的信任机制，导致网络难以吸引大量用户。

为了解决这一问题，最佳的方式是融合比特币已有的所有能力，来构建一个完全去中心化的 BTC Layer2 扩展方案。

BEVM 的 Taproot Consensus 就是这样一种创新性的设计。它将比特币的 Taproot 技术（Schnorr 签名和 MAST ），比特币 SPV 轻节点以及 BFT PoS 共识机制融合在一起，构建出了一个去中心化且高度一致性的 Layer2 网络。

利用比特币 SPV 轻节点技术使得 BEVM 能够轻量化且去中心化地同步比特币网络状态。同时 Taproot 技术和 BFT PoS 共识机制融合构建的门限签名网络，使得 BEVM 能够去中心化地将状态同步回比特币网络，从而形成了一个完全去中心化的双向通道。总之，Taproot Consensus 是在利用比特币网络固有安全性的基础上，实现了基于高度一致性 BFT PoS 共识的完全去中心化的 Layer2 扩展。

第二章：架构



2.1 组成部分概述

BEVM 的 Taproot consensus 由 Schnorr + Mast， Bitcoin SPV 和 Aura + Grandpa 三部分组成。

2.2 Schnorr + MAST 和 Bitcoin SPV

在 BEVM 系统中，每个验证者均持有一个用于 Schnorr 签名的 BTC 私钥。Schnorr 签名的特性使其能够实现高效的签名聚合，从而提高系统的安全性和效率。通过 Musig2 多签名方案生成的聚合公钥 P_{agg}，形成了一颗大型 MAST（ Merkle Abstract Syntax Tree ）树。

在 MAST 树的根哈希值生成后，验证者通过向 MAST 树生成的门限签名地址进行 BTC 转账和铭刻操作，实现 BTC 主网向 BEVM 网络提交数据的功能。同时每个验证者均作为 Bitcoin SPV（Simplified Payment Verification）轻节点，使其能够安全且无许可地同步 BTC 网络状态。

2.3 Schnorr + MAST 和 Aura + Grandpa

区块链是一种特殊的分布式网络，通过引入拜占庭容错机制（Byzantine Fault Tolerance， BFT），解决了去中心化信任的问题。区块链可被视为一种具备拜占庭容错特性的分布式网络。

在 BEVM 系统中，采用了 Aura 和 Grandpa 这两种先进的 PoS 共识机制，以保证网络的一致性和安全性。Aura 和 Grandpa 作为实现拜占庭容错的高级 PoS 共识协议，通过分布式协议确保网络节点的高度一致性。为了防止 PoS 共识中的验证者作恶，系统结合了 BTC 质押治理机制，利用 BTC 来保障 BEVM 网络的安全性。

引入 Aura 和 Grandpa 共识机制后，BEVM 系统中的分布式门限签名网络具备了拜占庭容错特性，形成了一个独特的 Layer2 区块链结构。该门限签名区块链不仅能够确保 BEVM 网络的安全性，还赋予了 BEVM 状态去中心化同步至 BTC 网络的能力。

2.4 tBTC 和 Taproot Consensus

Mezo 的底层技术结构是基于 tBTC 协议。tBTC 利用比特币多签构建了一个门限签名网络,这种结构相比传统分布式网络而言，具有较强的一致性。

然而,要实现真正去中心化且具有拜占庭容错特性的区块链方案，Mezo 还需要进一步将这个多签网络与 BFT PoS（拜占庭容错权益证明）共识机制相结合。

相比之下， Taproot Consensus 方案则采取了一种更为先进的设计。它通过结合 Schnorr 、MAST、Musig2 多签方案、比特币 SPV 轻节点以及 Aura 和 Grandpa 拜占庭容错共识机制，构建了一个高度一致性和安全的去中心化 Layer2 扩展方案。这种融合不仅提升了比特币网络的扩展性和可用性，还确保了 BEVM 网络的安全性和一致性。

第三章：门限签名

Taproot升级是比特币网络的一次重要改进，包括 Schnorr 签名（BIP 340）、Taproot（BIP 341）和 Tapscript（BIP 342）。主要目的是提升比特币的隐私性、效率和灵活性。Schnorr 签名允许将多个签名合并为一个，降低了交易费用和内存负担。Merkle Abstract Syntax Trees（MAST）将复杂的Bitcoin脚本隐藏在 Merkle 树的结构中，提高了交易的匿名性和隐私性。

MuSig2 是一种先进的多重签名方案，它使得多个参与者能够共同签署单一文件或交易，并确保最终的签名在外部观察时无法区分，看似是单一实体所生成。这种方案特别设计以在不安全的网络环境中实现多方之间的安全通信，提供了一种高效且安全的多签解决方案。

3.1 Schnorr 签名

- **生成公钥**：选取随机数d作为私钥，G为基点，那么公钥P：

$$P = d * G$$

- **生成签名**：选取随机数r作为 nonce，message 为签名消息，那么签名(r, s):

$$R = r * G$$

$$e = Hash(R, P, message)$$

$$s = r + e * d$$

- **聚合公钥和签名**: Schnorr 签名公钥和签名是可以聚合的

  已知：

$$d_{agg} = \sum_{i=1}^{n} d_i$$

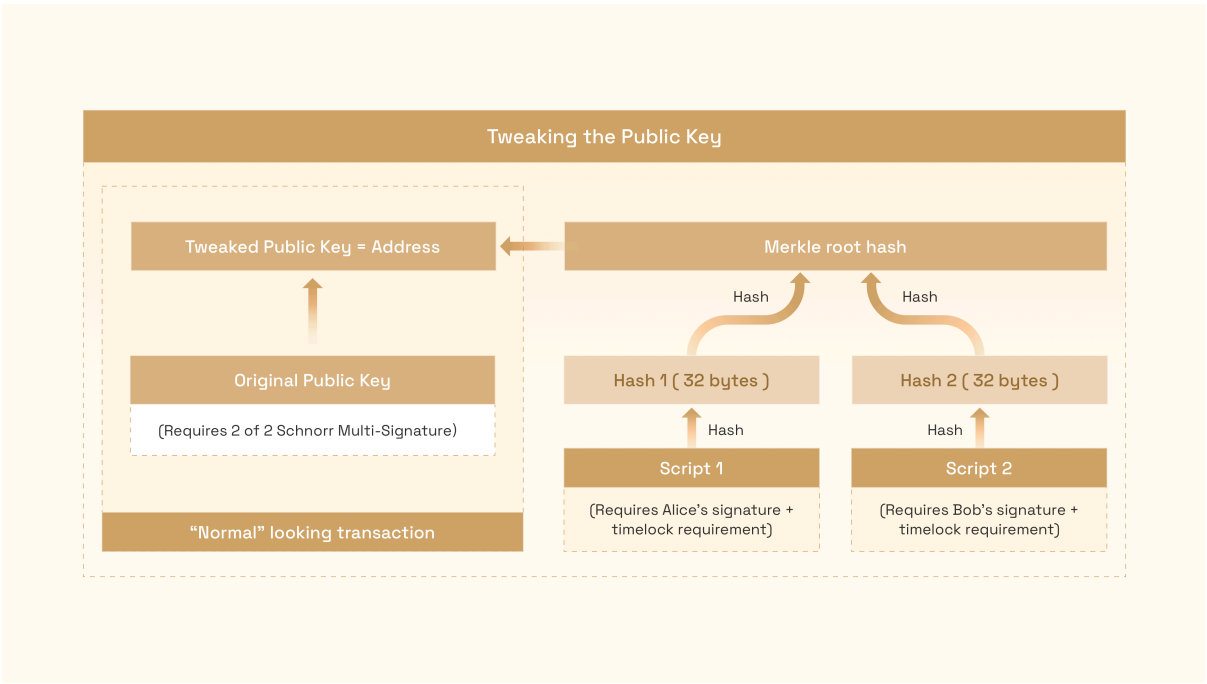
  那么：

$$P_{agg} = \sum_{i=1}^{n} P_i$$

$$s_{agg} = \sum_{i=1}^{n} s_i$$

Schnorr 签名是比特币 Taproot 升级的关键技术，取代了之前的 ECDSA 签名算法。从 Schnorr 签名的具体公式，很容易证明私钥（d）、公钥（P）和签名元素（r, s）都具备线性可加性。这种独特的属性允许将多个签名有效地聚合为一个单一签名，极大地减少了交易数据的大小，从而提高交易处理速度并降低费用。

3.2 Merkle Abstract Syntax Trees



MAST（Merkle Abstract Syntax Tree）是一种数据结构，结合了默克尔树和抽象语法树的优点，旨在优化比特币智能合约的隐私性和效率。MAST 允许将复杂的智能合约条件隐藏在 Merkle 树的结构中，只有在这些条件被触发时，相关的脚本部分(Script 1 或 Script 2)才会被揭露。

MAST 树可以用于组织和验证复杂的脚本和条件。构建 MAST 树的步骤如下：

- 脚本拆分：将复杂的智能合约脚本拆分为多个子脚本，每个子脚本代表一种可能的执行路径。
- 哈希计算：对每个子脚本计算其哈希值。设子脚本集合为 $S = \{S_1, S_2, \ldots, S_m\}$，其对应的哈希值集合为 $H = \{H(S_1), H(S_2), \ldots, H(S_m)\}$。
- 树节点构建：将哈希值集合作为叶节点，通过二叉树结构逐层计算父节点哈希值，最终生成根哈希值 H_{root}。该过程遵循默克尔树的构建方式。其中，H_{left} 和 H_{right} 分别为左右子节点的哈希值，$\|$ 表示字符串连接操作。

$$H_{\text{parent}} = H(H_{\text{left}} \| H_{\text{right}})$$

- MAST 根哈希：最终生成的根哈希值 H_{root} 作为 MAST 树的根哈希，用于代表整个智能合约的状态。

3.3 MuSig2

Musig2 是一种多重签名方案，是 MuSig 签名方案的一个变体。MuSig2 允许多个签名者从他们各自的私钥中创建一个聚合公钥，然后共同为该公钥创建一个有效签名，通过这个方式创建的聚合公钥与其他公钥是无法区分的。MuSig2 是一种简单且高度实用的两轮多重签名方案，具有优势： i) 在并发签名会话下是安全的，ii) 支持密钥聚合，iii) 输出普通 Schnorr 签名，iv) 只需要两轮通信，v) 具有与普通 Schnorr 签名相似的签名者复杂性。

$\text{Game CORRECT}_{\Sigma,m,n,j}(\lambda)$

$par \leftarrow \mathsf{Setup}(1^\lambda)$
for $i := 1 \ldots n$ **do**
  $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}()$
  $(out_i, state_i) \leftarrow \mathsf{Sign}()$
$out := \mathsf{SignAgg}(out_1, \ldots, out_n)$
for $i := 1 \ldots n$ **do**
  $(out'_i, state'_i) \leftarrow \mathsf{Sign}'(state_i, out, sk_i, m, (pk_1, \ldots, pk_{i-1}, pk_{i+1}, \ldots, pk_n))$
$out' := \mathsf{SignAgg}'(out'_1, \ldots, out'_n)$
$\sigma \leftarrow \mathsf{Sign}''(state'_j, out')$
$\widetilde{pk} := \mathsf{KeyAgg}(pk_1, \ldots, pk_n)$
return $\mathsf{Ver}(\widetilde{pk}, m, \sigma)$

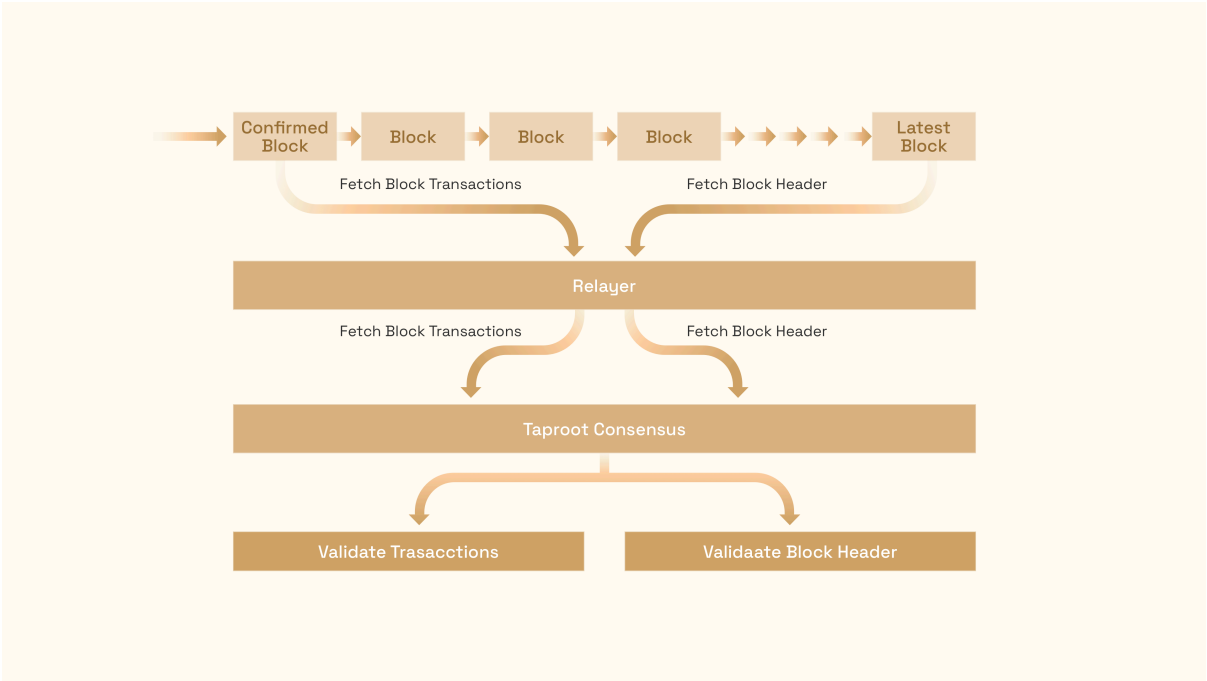
MuSig2 签名过程涉及两个主要阶段，首先是密钥和临时公私钥的生成，其次是通过两轮通信来完成签名的聚合。在开始阶段，每个参与者使用 `KeyGen()` 函数生成自己的公私钥对，并通过 `Sign()` 函数创建一个临时的公私钥对（即 nonce ）。这个临时公钥（out_i）随后在第一轮通信中被发送给其他参与者。一旦收到其他所有参与者的临时公钥，每个人可以通过`SignAgg()`和`Sign'()`函数生成自己的签名碎片（out'_i）。这些签名碎片在第二轮通信中被交换。在获取了所有其他参与者的签名碎片后，每个人使用`SignAgg'()`和`Sign''()`函数将这些碎片组合，生成最终的统一签名（σ）。

第四章： BTC SPV 轻节点

在比特币网络中，BTC Simplified Payment Verification（SPV）轻节点引入了一种高效的机制，允许在不下载完整区块链的情况下验证 BTC 交易。这一特性使得 Taproot Consensus 能够在完全去中心化的环境下，无需任何许可，同步 BTC 状态。通过仅同步相关交易，SPV 轻节点在不牺牲安全性的前提下，显著降低了存储需求，提供了一个既轻便又安全的 BTC 状态同步机制。

4.1 工作原理

由无许可的 Relayer 自动从 BTC 网络中获取最新的块头，并将已确认块中的交易分别推送给节点，根据 SPV 轻节点验证规则实现对交易和区块进行验证，完成和BTC的交互。其中区块头包含了足够的信息来进行交易验证，如前一个区块的哈希值、时间戳、难度证明（nonce）和 Merkle 根。这使得节点可以直接存储同步块头来确认交易的有效性而无需存储整个区块的内容。



4.2 验证区块头

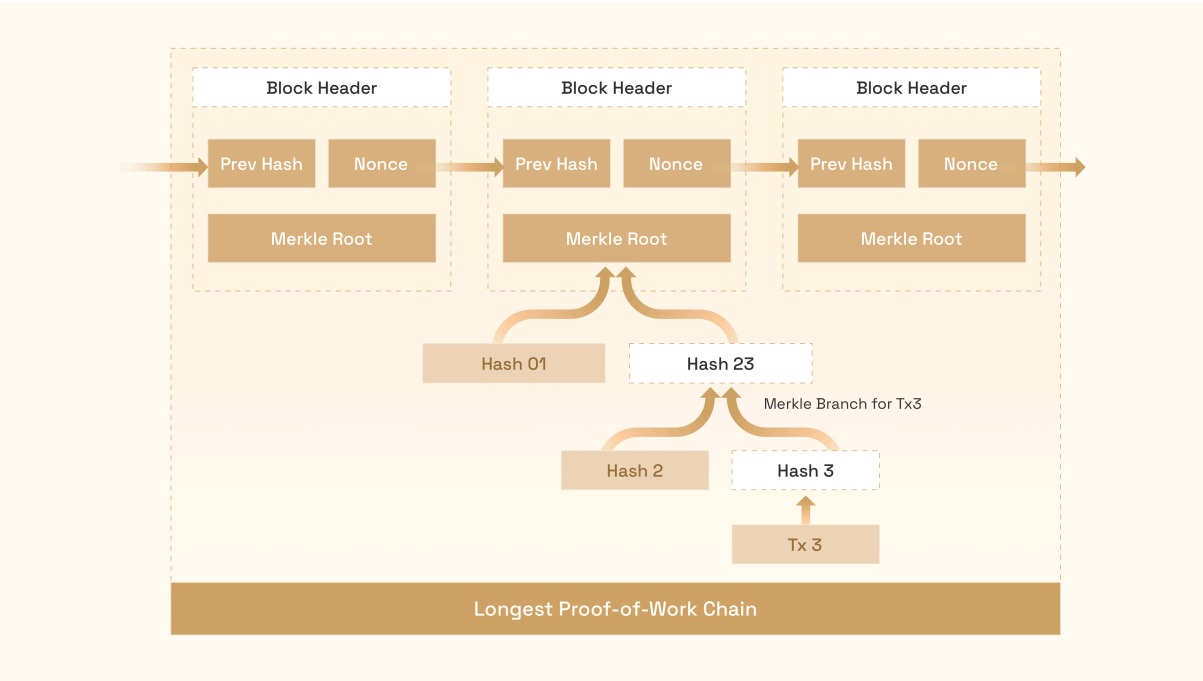
验证区块头涉及检查区块头信息的有效性和完整性。

```
// A Bitcoin header is a 80-byte structure that contains the block metadata.
// The structure is defined in the Bitcoin protocol as follows:
// ------------------------------------------------------------------------------------
// | Name            | Type    | Bytes  | Description                                 |
// ------------------------------------------------------------------------------------
// | version         | int32_t | 4      | The block version number                    |
// | previous_hash   | char[32]| 32     | The hash of the previous block              |
// | merkle_root     | char[32]| 32     | The root of the merkle tree of the transactions |
// | time            | uint32_t| 4      | The time of the block                       |
// | bits            | uint32_t| 4      | The target difficulty of the block          |
// | nonce           | uint32_t| 4      | The nonce used to find the proof of work    |
// ------------------------------------------------------------------------------------

// Bitcoin Header #840000
// https://mempool.space/api/block/00000000000000000000320283a032748cef8227873ff4872689bf23f1cda83a5/header
// Decode to struct
0x2a5fe000 ..................................................................... version
0x00000000000000000000172014ba58d66455762add0512355ad651207918494ab ............. previous_hash
0x031b417c3a1828ddf3d6527fc210daafcc9218e81f98257f88d4d43bd7a5894f ............. merkle_root
0x662307b7 ..................................................................... time
0x17034219 ..................................................................... bits
0xea63987d ..................................................................... nonce

// Check difficulty target
// need retarget every 2016 blocks
// new_target = old_target * ((time_span_last_block - time_span_first_block)/ (2016 * 10 * 60))
// require new_target == bits

// Check pow valid
bits = 0x17034219
coefficient = 0x034219 = 213529
exponent = 0x17 = 23

// target = coefficient * 2^(8 * (exponent - 3))
target = 213529 * 2^(8 * (23 - 3)) = 3120729831176303692211146186450751969041116199691223 04

block_hash = 0x00000000000000000000320283a032748cef8227873ff4872689bf23f1cda83a5
// require block_hash <= target
```

1. **下载区块头，检查前一个区块的哈希**：获取区块头信息，包括版本号、前一个区块的哈希、Merkle 根、时间戳、难度目标和随机数（nonce）。确认当前区块头中包含的前一个区块的哈希值是否正确，确保当前区块正确链接到前一个区块。
2. **验证难度目标**：检查当前区块的难度目标是否符合比特币网络的要求，确保计算难度与网络标准一致。
3. **验证工作量证明（PoW）**：通过检查区块头的哈希值，确保其小于等于当前的难度目标，验证工作量证明的有效性。

4.3 验证交易在区块中

验证交易涉及检查交易是否有效地包含在区块中。下面是其过程：



1. 构建目标交易的 Merkle 路径。
2. 使用 Merkle 路径和交易哈希计算 Merkle 根，并与区块头中的 Merkle 根对比。
3. 确认交易格式和签名有效性，确保交易未被双重支付。

验证的过程利用了比特币区块链的 Merkle 树结构，确保了验证的高效性和准确性。这种方法广泛应用于轻节点和其他需要验证交易存在性的场景。

第五章： BEVM 的 BFT pos 共识（Aura+ Grandpa）

BEVM 使用了Aura（Authority Round）和 Grandpa（GHOST-based Recursive Ancestor Deriving Prefix Agreement）这两种共识机制的组合，增强了区块链网络的效率、安全性和最终性。通过结合拜占庭容错（BFT）机制，进一步提升了网络的可靠性和抗攻击能力。

5.1 Aura共识机制

Aura（Authority Round）是一种基于权威证明（PoA）的共识算法，主要用于出块。其工作原理如下：

- **轮流出块**：Aura 采用基于权限的轮转调度机制，在每个时间槽（slot）中，预先确定的一组权限节点（验证者）轮流生成区块。每个验证者在其轮到的时间槽生成一个新区块。出块者根据以下公式确定：

$$Validator_{current} = Validator_{(slot \mod N)}$$

其中，$Validator_{current}$是当前出块者，*N* 是验证者的总数，*slot* 是当前的时间槽编号。

- **固定间隔**：区块生成时间是固定的（例如，每6秒一个区块），这使得出块过程更加可预测和稳定。具体来说：

$$T_{block} = T_{slot}$$

其中，T_{block}是区块生成时间，T_{slot}是时间槽的固定长度。

- **快速出块**：由于出块者是预先确定的，Aura能够实现快速且高效的区块生成。

Aura的主要优势在于其简洁性和高效性，适用于需要快速出块的区块链网络。

5.2 Grandpa 共识机制

Grandpa（GHOST-based Recursive ANcestor Deriving Prefix Agreement）是一种用于区块最终确定性的共识算法。其工作原理如下：

- **多轮投票**：Grandpa 通过多轮投票机制，使网络中的验证者对区块链的某个前缀（prefix）达成一致。每个验证者在每一轮中都对其认为最优的区块链进行投票，直到网络中超过三分之二的节点达成一致。投票结果统计公式为：

$$\text{VoteCount}(b) = \sum_{v \in V} \text{vote}(v, b)$$

其中，*V* 为所有验证节点的集合，*b* 为候选区块集合，$\text{vote}(v, b)$ 表示验证节点 *v* 对区块 *b* 的投票。

- **区块确认**：一旦超过三分之二的验证者对某个区块达成共识，该区块及其所有祖先区块都被认为是最终确定的，不可逆转。共识达成条件为：

$$\text{VoteCount}(b) > \frac{2}{3}N$$

其中，*N* 为验证节点总数。

- **高效性和安全性**：Grandpa 结合了拜占庭容错算法，确保在存在恶意节点的情况下，系统仍能稳定运行。

Grandpa 的主要优势在于其高效的最终确定性，即使在网络分叉的情况下也能快速达成共识。

第六章： BTC SPV 轻节点，门限签名和 BFT pos 共识的融合

BEVM 网络是 BTC SPV轻节点，门限签名和 BFT pos 共识三者有效的融合，包括：

6.1 生成门限签名地址

假设验证者集合为 $V = \{V_1, V_2, \ldots, V_n\}$，每个验证者 V_i 具有公钥 P_i。其中任意 m 个结点利用Musig2 多签方案生成聚合公钥P_{agg}，将其作为 MAST 树的 Script 脚本，构建出一颗大型 MAST 树，从而生成门限签名地址。所有的验证者集合通过这种 MAST 结构创建了一个稳健且高效的门限签名网络，允许任何n个验证者中的 *m* 个完成通过 Musig2 的两轮签名机制完成签名和执行脚本。

6.2 同步 BTC 主网

BTC SPV 的轻便性和高效性使得在 BEVM 中，每一个验证者作为一个独立 BTC SPV 节点成为可能。所有的验证者集合共同构建了一个BTC SRV分布式网络。由于BTC SPV的特性，使得 BEVM 网络天然具有安全无许可同步 BTC 网络状态的能力。通过向门限签名地址发送转账或铭刻交易，实现 BEVM 网络同步 BTC 主网数据。

6.3 提交 BEVM 数据

BEVM 将门限签名技术与 Aura + Grandpa 这种先进的拜占庭容错共识机制相结合,构建了一个高度一致性和安全性的去中心化分布式网络。引入这种拜占庭容错共识之后，BEVM 的门限签名分布式网络得以实现真正的去中心化。当用户在 BEVM 发起提现交易时，验证者集合中其中的 m 个成员将使用Musig2多签方案进行通信签名，最终完成 BEVM 向 BTC 网络的数据提交。

6.4 BTC治理

门限签名技术与拜占庭容错（BFT）权益证明（PoS）共识机制的融合是通过一个多层次的密钥管理和治理结构来实现，该结构确保了系统的去中心化和安全性。

6.4.1 多层次密钥管理

在这种架构中，验证节点的选举和治理通过PoS治理系统进行，涉及多个关键密钥的管理，包括治理密钥、出块密钥和门限签名密钥。这些密钥的设置和管理均通过无许可的用户投票来实现。

1. **治理密钥（Governance Key）**：用于参与网络治理和验证节点的选举。治理密钥允许持有者对网络参数和规则进行投票，从而影响网络的整体治理结构。

2. **出块密钥（Block Key）**：专门用于生成新区块和验证交易。出块密钥确保了验证节点在出块过程中的权威性和安全性。
3. **门限签名密钥（Threshold Signature Key）**：用于实现分布式的门限签名方案。门限签名密钥通过多个验证节点的合作生成和管理，确保在一定数量的节点同意的情况下才能执行签名操作。这种机制增强了系统的安全性和容错能力，防止单点故障和恶意行为。

6.4.2 治理角色

- **普通用户（Stakers）**：持有BTC的用户可以将比特币质押到网络中，参与验证节点的竞选和治理决策。通过质押BTC，用户获得治理代币（Governance Tokens），这些代币赋予他们投票权和提案权。
- **验证节点（Validators）**：验证节点通过竞选获得，持有治理密钥和出块密钥，并参与门限签名操作。验证节点的选举和管理通过PoS治理系统进行，确保其去中心化和安全性。
- **理事会（Council）**：由治理代币持有者选举产生，负责提出和审核提案，确保治理过程的高效和合理。理事会成员持有治理密钥，并对关键决策具有更高的投票权重。
- **技术委员会（Technical Committee）**：由理事会选举产生，负责紧急情况下的技术决策和升级。技术委员会成员具有快速决策权，以应对突发事件。

6.4.3 治理流程

- **提案阶段**：任何持有治理代币的用户都可以提出治理提案。这些提案可以涉及网络参数的调整、验证节点的管理、关键密钥的设置等。
- **审核阶段**：提案提交后，理事会对其进行审核。理事会成员可以修改、接受或拒绝提案。通过审核的提案进入投票阶段。
- **投票阶段**：所有治理代币持有者都可以对提案进行投票。投票采用加权投票机制，持有更多治理代币的用户投票权重更大。
- **执行阶段**：通过投票的提案进入执行阶段。执行涉及相应的技术操作，如密钥的生成和分发、网络参数的调整等。

6.4.4 与BTC质押的结合

BFT PoS共识的安全性取决于参与者的质押资金。为了增强网络的安全性和可靠性，BEVM 引入了比特币质押治理机制，要求验证节点将一定数量的BTC质押到网络中。

- **质押和奖励**：持有BTC的用户可以将比特币质押到网络中，参与验证节点的竞选和治理决策。通过质押BTC，用户可以获得区块奖励和交易费用作为回报。
- **治理权利**：质押BTC的用户还享有治理代币的治理权利，包括提出提案和参与投票。支持通过的提案的质押者可获得额外奖励，而提交恶意提案的质押者将失去质押的比特币作为惩罚。
- **安全保障**：一旦发生恶意行为，这些质押资金将被罚没，有效遏制了作恶动机。因此，质押的BTC数量越多，BEVM 网络就越安全可靠。

总的来说，BEVM 通过融合多项先进的比特币技术,打造了一个去中心化、安全且高效的比特币 Layer2 解决方案。这不仅保护了 BTC 资产的安全,也为比特币生态的未来发展注入了新的活力。

参考文献

[1] Nakamoto, S. (2008, October 31). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf

[2] Wuille, P., Nick, J., & Ruffing, T. (2020). Schnorr Signatures for secp256k1. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki

[3] Wuille, P., Nick, J., & Ruffing, T. (2020). Taproot: SegWit version 1 spending rules. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki

[4] Wuille, P., Nick, J., & Ruffing, T. (2020). Validation of Taproot scripts. Retrieved from https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki

[5] Poelstra, A., Ruffing, T., & Seurin, Y. (2020). MuSig2: Simple Two-Round Schnorr Multisignatures. Retrieved from https://eprint.iacr.org/2020/1261

[6] Parity Technologies. (n.d.). Aura: Authority Round Consensus Algorithm. Retrieved from https://openethereum.github.io/Aura

[7] Parity Technologies. (n.d.). GRANDPA: A Byzantine Finality Gadget. Retrieved from https://github.com/w3f/consensus/blob/master/pdf/grandpa.pdf