官网：https://babylonchain.io/
开发者文档：https://docs.babylonchain.io/
文档：https://github.com/babylonchain/babylon/blob/dev/docs/architecture.md
github：https://github.com/babylonchain；
博客：https://babylonchain.io/blog
浏览器：https://babylonscan.io/

Babylon 的比特币质押协议：无需信任、安全且快速

PoS 安全的关键是，如果质押者攻击 PoS 链，则削减质押。
Babylon 使用**提取一次性签名 (EOTS)** 等尖端加密技术，将可削减的 PoS 攻击转换为可花费的比特币 UTXO 进行销毁。
这种原语与比特币的本地时间锁定一起，允许比特币持有者通过简单地将比特币锁定在比特币网络上来质押他们的比特币并参与 PoS 安全。

不涉及第三方，只要质押者不攻击 PoS 链，比特币就是安全的。
Babylon 还应用了最先进的**比特币时间戳协议，以实现比特币和 PoS 链之间的紧密同步，从而允许快速解绑所质押的比特币。**

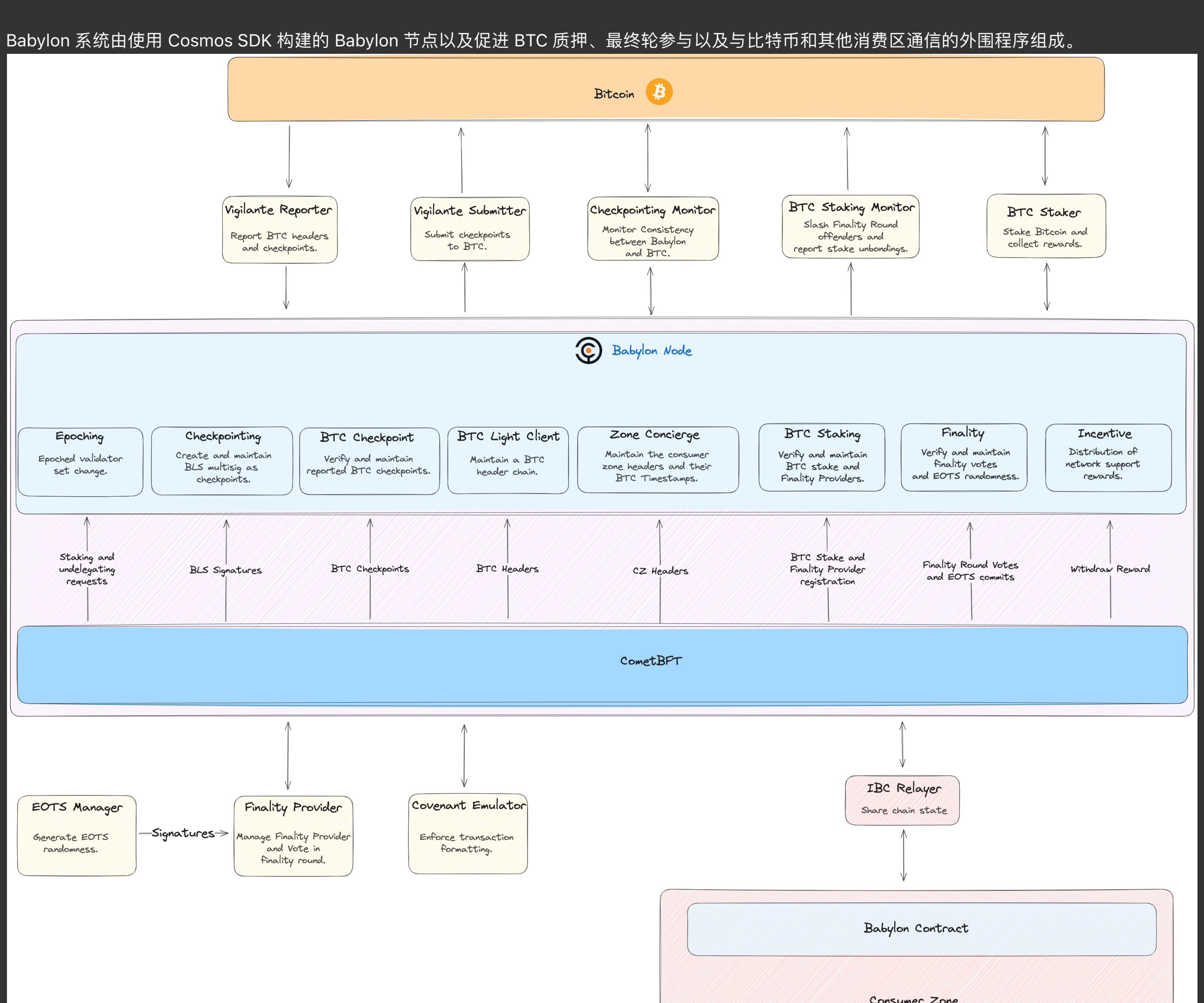
巴比伦的比特币质押协议：模块化且可扩展的双边市场

Babylon 的比特币质押协议创建了一个双边市场，并充当市场的控制平面。
比特币持有者可以安全地锁定他们的比特币，并选择质押的 PoS 链并从中赚取收益。
PoS 链和 dApp 可以选择采用比特币支持的安全性，并享受高安全性、健康的经济性和更广泛的采用。
该协议是模块化的，可以保护任何 PoS 链。它还可以为比特币持有者提供可扩展的重新抵押。

比特币时间戳协议
我们的起源可以追溯到一篇关于比特币安全性的变革性研究论文，
这是一篇由我们的联合创始人 David Tse 和 Fisher Yu 以及合著者共同撰写的原创作品。https://arxiv.org/pdf/2207.08392。

巴比伦建筑

Babylon 系统由使用 Cosmos SDK 构建的 Babylon 节点以及促进 BTC 质押、最终轮参与以及与比特币和其他消费区通信的外围程序组成。



一：巴比伦节点模块

模块使用指南：https://docs.babylonchain.io/docs/developer-guides/modules/overview

1、划时代:**epoching**

巴比伦区块链被划分为由参数化数量的区块组成的纪元。在每个时期内，验证者集不会改变。

这样，Babylon 需要每个纪元而不是每个区块都有一个检查点，从而降低了检查点成本。

纪元模块通过延迟影响验证器集的事务的执行来实现这一点，该验证器设置为每个纪元的最后一个块。

babylon禁用了原生staking模块的即验证器集更新机制和 21 天解绑机制：

为了禁用这两项功能，Babylon 禁用了质押模块的EndBlocker功能，该功能在区块结束时更新验证器集并解除成熟验证器的绑定。

相反，在一个 epoch 结束时，epoching 模块将调用质押模块的功能来更新验证器集。

此外，在一个 epoch 已检查点到比特币后，epoching 模块将调用质押模块的功能来解除成熟验证器的绑定。

```
app.ModuleManager.SetOrderEndBlockers(crisistypes.ModuleName, govtypes.ModuleName, stakingtypes.ModuleName,
        capabilitytypes.ModuleName, authtypes.ModuleName, banktypes.ModuleName, distrtypes.ModuleName,
        slashingtypes.ModuleName, minttypes.ModuleName,
        genutiltypes.ModuleName, evidencetypes.ModuleName, authz.ModuleName,
        feegrant.ModuleName,
        paramstypes.ModuleName, upgradetypes.ModuleName, vestingtypes.ModuleName, consensusparamtypes.ModuleName,
    )
    // Babylon does not want EndBlock processing in staking
    app.ModuleManager.OrderEndBlockers = append(app.ModuleManager.OrderEndBlockers[:2],
app.ModuleManager.OrderEndBlockers[2+1:]...) // remove stakingtypes.ModuleName
```

为了使验证器集在周期中保持不变，周期模块会通过 AnteHandler 拦截并拒绝影响验证器集的质押相关消息，

而是定义它们的包装版本，并在周期结束时将其包形式转发给质押模块。

回想一下，Cosmos SDK 模块包含用于交易消息的 protobuf 文件。在质押模块中，这些消息包括

- MsgCreateValidator用于创建新的验证器
- MsgEditValidator用于编辑现有的验证器
- MsgDelegate将代币从委托人委托给验证人
- MsgBeginRedelegate用于将代币从委托人和源验证者重新委托给目标验证者。
- MsgUndelegate用于从委托人和验证者取消委托。
- MsgCancelUnbondingDelegation取消委托人的解除委托

在这些消息中，MsgCreateValidator、MsgDelegate、MsgBeginRedelegate和MsgUndelegate会影响验证器集。
由于 Babylon 要求验证器集在一个时期内保持不变，因此它必须避免在时期中间处理这些消息。
为此，epoching 模块定义了一个 AnteHandler 来拒绝这些消息。
相反，它在 epoching 模块中为它们定义了包装版本：
MsgWrappedCreateValidator、MsgWrappedDelegate、MsgWrappedBeginRedelegate和MsgWrappedUndelegate。
epoching 模块随时接收这些消息，但只会在每个时期结束时处理它们。

将包装的消息延迟到Epoch

纪元模块将在每个纪元结束时处理包装的质押相关消息。为此，纪元模块为每个纪元维护一个消息队列。

对于每条包装的消息，纪元模块都会执行基本的健全性检查，然后将消息排入消息队列。

当纪元结束时，纪元模块将解开排队的消息并将其转发给质押模块。

因此，质押模块在每个纪元结束时接收并处理质押相关消息，从而执行验证器集更新。

比特币辅助接触绑定

比特币辅助解绑是一种机制，即只有当包含相关解绑请求的区块由比特币检查点时，解绑验证者或委托人才会被解绑。

该机制对于实现可削减安全性是必要的，如果验证者执行安全攻击，则必须被削减。

Babylon 通过在某个时期完成检查点时调用质押模块来实现比特币辅助解绑机制。
具体来说，质押模块ApplyMatureUnbondings负责识别和解绑已解绑 21 天的成熟验证者和委托，并在每个区块上调用。
Babylon 已禁用ApplyMatureUnbondings每个区块的调用，并实现时期的状态管理。当某个时期完成时，时期模块将调用ApplyMatureUnbondings以在该时期结束之前解绑所有解绑验证者和委托。

2、BTC轻客户端：**btclightclient**

BTC轻客户端模块接收Vigilante Reporter报告的比特币标头，并根据比特币的PoW规则维护一条BTC标头链。

它公开了有关规范比特币链、标头深度以及比特币交易的包含证据是否有效的信息。

3、比特币检查点：**btccheckpoint**

BTC Checkpoint 模块验证 Vigilante Reporter 报告的 Babylon BTC 检查点，

并根据 BTC Light Client 模块根据检查点的深度向 Checkpointing 模块提供这些检查点的确认状态。

4、检查点：**checkpointing**

检查点模块负责创建提交给比特币的巴比伦检查点并维护其确认状态。

它 为要设置检查点的每个块收集验证者的 BLS 签名，并将它们聚合成 BLS 多重签名以包含在比特币检查点中。

每个检查点的确认状态由从 BTC 检查点模块检索的比特币检查点包含信息确定。

5、礼宾区：**zoneconcierge**

Zone Concierge 模块从连接的IBC 轻客户端中提取经过验证的 Consumer Zone 标头，

并根据承载它们的 Babylon 交易的比特币确认状态来维护其比特币确认状态。

它通过IBC连接将可验证的证据将比特币确认状态传达给消费者区 。

6、BTC 质押：**btcstaking**

BTC Stake 模块是 BTC Stake 协议的簿记员。它负责验证和激活 BTC 质押请求并维护活跃的最终性提供商集。

它与 BTC 轻客户端模块通信，以获取质押请求的确认状态，并从 BTC 质押监视器接收有关按需解锁质押的通知。

7、最终性：**finality**

Finality 模块负责最终确定 CometBFT 共识产生的区块。

它接收并验证来自最终性提供者的最终性轮投票，如果有足够的投票权，则该区块被视为最终确定。

每个最终性提供商的投票权基于从 BTC Stake 模块检索的比特币权益。

最终性投票是使用 可提取一次性签名（EOTS）进行的，并使用最终性提供者承诺的公共随机性进行验证。

8、激励：**incentive**

激励模块消耗了巴比伦质押者奖励的一定比例，并将其作为奖励分配给比特币质押者和治安维护者。

二：Vigilantes：

义务警员共有四个治安维护计划：

1、提交者-submitter：向比特币提交巴比伦检查点。https://github.com/babylonchain/vigilante/tree/dev/submitter

2、记者-reporter：向巴比伦报告比特币头段和巴比伦检查点。https://github.com/babylonchain/vigilante/tree/dev/reporter
 * 使用 BTC 节点同步最新的 BTC 区块
 * 从 BTC 区块中提取区块头和检查点
 * 将标头和检查点转发到 Babylon 节点
 * 检测并报告 BTC 区块链与 Babylon BTCLightclient 标头链之间的不一致
 * 检测并报告 BTC 区块中检查点深度为 w 但 Babylon 尚未包含其 k 深度证明的停滞攻击

3、BTC 时间戳监视器-BTC timestamping monitor：监视巴比伦检查点站的审查制度。

4、**BTC 质押追踪器-BTC staking tracker：监控 BTC 委托的早期解绑情况并削减对抗性最终性提供者。**

2.1、私用提交者:Vigilante Submitter

一个独立程序，利用比特币脚本代码将巴比伦检查点作为嵌入数据的比特币交易提交给比特币OP_RETURN。

2.2、治安记者：Vigilante Reporter

一个独立的程序，扫描比特币分类账中的比特币标头和巴比伦检查点，并使用巴比伦交易将它们报告回巴比伦。

三：Monitors-监控

监控程序套件负责监控Babylon状态与比特币之间的一致性。

3.1、检查点监视器：Checkpointing Monitor

一个独立的程序，用于监视：

- 比特币规范链和比特币头链之间的一致性由Babylon的BTC Light客户端模块维护。
- 及时将巴比伦的比特币检查点信息纳入巴比伦账本。

3.2、BTC 质押监控：BTC Staking Monitor

一个独立的程序，用于监视：

- 在比特币账本上执行 BTC 质押按需解绑交易，以通知巴比伦相关信息。
- 在最终性提供者双重投票的情况下执行 BTC Stake 削减交易。在不执行的情况下，监视器提取最终性提供者的私钥并执行削减。
- 由最终性提供者发起的选择性削减攻击的执行。在这种情况下，监视器会提取最终性提供者的私钥并删除它们。

四：BTC 质押计划

BTC 质押程序套件涉及启用比特币质押者和最终性提供者功能的组件，同时确保他们遵守协议。

4.1、BTC 质押者

比特币持有者可以通过创建一组比特币交易来抵押他们的比特币，将它们添加到比特币账本中，然后通知巴比伦他们的抵押。

之后，他们还可以在股份到期时按需解锁或提取资金。开发了以下一组独立程序来实现这些功能：

- BTC Staker Daemon：连接比特币钱包和Babylon的守护程序。
- BTC Staker Dashboard：连接到比特币钱包扩展和 Babylon API 的 Web 应用程序。只能用于测试目的。
- 钱包集成（待定）

4.2、确定性提供者:https://github.com/babylonchain/finality-provider

一个独立的程序，允许注册和维护最终确定性提供者。

最终性提供者是否包含在活动集中，提交 可提取一次性签名（EOTS） 公共随机性，并提交区块的最终性投票。

最终投票是通过连接到独立的 EOTS 管理器守护进程创建的，该守护进程负责安全维护最终提供者的私钥。

4.3、契约模拟器:https://github.com/babylonchain/covenant-emulator

盟约模拟委员会成员使用的独立程序。它通过验证待处理的质押请求、验证其内容并提交必要的签名来模拟契约功能。

五：消费区

5.1、IBC中继器

IBC 中继器维护 Babylon 和其他消费者区 (CZ) 之间的IBC 协议连接。

它负责更新 Babylon 账本内的 CZ 轻客户端，以启用检查点并将检查点信息传播到 CZ 内部署的 Babylon 智能合约。

有不同的 IBC 中继器实现可以实现此功能。最为显著地：

- Cosmos Relayer：用 Go 编写的功能齐全的中继器。
- Babylon Relayer：Cosmos Relayer 的包装器，可以维持单向 IBC 连接。建议在Consumer Zone未部署Babylon智能合约时使用。
- Hermes Relayer：用 Rust 编写的功能齐全的中继器。

5.2、巴比伦契约

旨在部署在消费者区的CosmWasm智能合约。它支持比特币检查点功能，而无需在消费者区的代码库中引入侵入性更改。

基于比特币检查点功能，消费者区可以根据比特币账本中包含的检查点做出决策（例如执行BTC辅助的解除绑定请求）。

六：babylon链核心模块：

1、Babylon modules

EpochingKeeper   epochingkeeper.Keeper
BTCLightClientKeeper btclightclientkeeper.Keeper
BtcCheckpointKeeper btccheckpointkeeper.Keeper
CheckpointingKeeper checkpointingkeeper.Keeper
MonitorKeeper   monitorkeeper.Keeper

2、IBC-related modules

IBCKeeper   *ibckeeper.Keeper
IBCFeeKeeper   ibcfeekeeper.Keeper
TransferKeeper   ibctransferkeeper.Keeper
ZoneConciergeKeeper zckeeper.Keeper

3、BTC staking related modules

BTCStakingKeeper btcstakingkeeper.Keeper
FinalityKeeper   finalitykeeper.Keeper

4、tokenomics-related modules
IncentiveKeeper incentivekeeper.Keeper

Zone Concierge 模块负责从其他 PoS 区块链生成标头的 BTC 时间戳。

这些BTC时间戳允许PoS区块链与Babylon集成以实现比特币安全，即分叉PoS区块链与分叉比特币一样困难。

Zone Concierge 模块利用 IBC 协议接收 PoS 区块链的标头，并为其提供有关其时间戳的简洁且可证明的信息。

PoS 区块链的集成分为两个阶段：

- **第一阶段集成：** Babylon 通过 MsgUpdateClientIBC 轻客户端协议中的标准消息接收 PoS 区块链标头，为其添加时间戳，并充当 PoS 区块链的规范链预言机。Babylonscan展示了具有第一阶段集成的 PoS 区块链。
- **第 2 阶段集成：** 除了第 1 阶段之外，第 2 阶段允许 PoS 区块链通过 IBC 通道从 Babylon 接收 BTC 时间戳，以便 PoS 区块链可以使用 BTC 时间戳来检测和解决分叉，以及其他用例，例如比特币辅助的快速解绑。

七：启动babylon的验证节点：
连接：https://docs.babylonchain.io/docs/user-guides/btc-staking-testnet/become-validator
八：最终确定性提供商
8.1 ETOS manager：
EOTS 守护进程负责管理 EOTS 密钥、产生 EOTS 随机性以及使用它们产生 EOTS 签名。

注意： EOTS 代表可提取一次性签名。您可以在Babylon BTC Staking Litepaper中阅读更多相关信息。
简而言之，EOTS 管理员会生成 EOTS 公有/私有随机性对。
最终性提供者会将这些对的公共部分提交给 Babylon，用于他们打算为其提供最终性签名的每个未来区块高度。
如果最终性提供者在同一高度上为两个不同的区块投票，他们将不得不重复使用相同的私有随机性，
这将导致其底层私钥被暴露，从而导致他们和所有委托人被削减。

EOTS 管理器负责以下操作：

1、EOTS 密钥管理：
  使用BIP-340标准为给定的最终性提供者生成Schnorr密钥对 。
  在内部 Cosmos 密钥环中保留生成的密钥对。
2、随机性生成：
  根据 EOTS 密钥、chainID 和区块高度生成 EOTS 随机性对列表。
  随机性是确定性生成的，并与特定参数相关。
3、签名生成：
  使用最终性提供者的私钥和指定高度的给定链的相应秘密随机性来签署 EOTS。
  使用最终性提供者的私钥签署 Schnorr 签名。
  EOTS 管理器作为该eotsd工具控制的守护进程运行。

3.启动EOTS守护进程：
您可以使用以下命令启动 EOTS 守护进程：

eotsd start --home /path/to/eotsd/home

如果--home未指定标志，则将使用默认主位置。

eotsd.conf这将在字段下指定的地址启动 EOTS rpc 服务器RpcListener，该地址默认设置为127.0.0.1:12582。您可以在配置文件中更改此值或覆盖此值并使用标志指定自定义地址--rpc-listener。

eotsd start

2024-02-08T17:59:11.467212Z info RPC server listening {"address": "127.0.0.1:12582"}
2024-02-08T17:59:11.467660Z info EOTS Manager Daemon is fully active!

可以使用标志查看所有可用的 cli 选项--help。这些选项也可以在配置文件中设置。

注意：建议eotsd在单独的机器或网段上运行守护程序以增强安全性。这有助于隔离密钥管理功能并减少潜在的攻击面。
您可以 EOTSManagerAddress在 finality 提供程序的配置文件中编辑以引用正在运行的机器的地址eotsd。
8.2 最终确定性提供者
最终性提供者守护进程负责监控新的 Babylon 区块、为其打算提供最终性签名的区块承诺公共随机性以及提交最终性签名。

守护进程可以为多个最终性提供者管理和执行以下操作：
1、创建和注册：创建并注册 Babylon 的最终性提供者。
2、EOTS 随机性承诺：守护进程监控 Babylon 链，并为每个最终性提供者打算投票的每个 Babylon 区块提交 EOTS 公共随机性。
  提交间隔可以在配置中指定。EOTS 公共随机性是通过最终性提供者守护进程与 EOTS守护进程的连接来检索的。
3、最终投票提交：守护进程监视巴比伦链并为每个维护的最终性提供者承诺投票的每个区块产生最终投票。

守护进程由工具控制fpd。fpcli工具实现与守护进程交互的命令。

5. 创建并注册最终
我们通过 fpcli create-finality-providerorfpcli cfp命令创建一个 finality 提供程序实例。创建的实例与一个 BTC 公钥（作为其唯一标识符）和一个 Babylon 账户相关联，权益奖励将直接发送到该账户。请注意，如果--key-name未指定标志，则将使用步骤Key中指定的配置字段。

```
fpcli create-finality-provider \
    --key-name my-finality-provider \
    --chain-id bbn-test-3 \
    --commission 0.05 \
    --moniker "your_finality_provider_name_here" \
    --identity "your_keybase_username_or_identifier_here" \
    --website "your_website_here" \
    --security-contact "your_security_contact_email_here" \
    --details "your_other_details_here"

{
    "babylon_pk_hex": "02face5996b2792114677604ec9dfad4fe66eeace3df92dab834754add5bdd7077",
    "btc_pk_hex": "d0fc4db48643fbb4339dc4bbf15f272411716b0d60f18bdfeb3861544bf5ef63",
    "description": {
        "moniker": "my-name"
    },
    "status": "CREATED"
}
```

fpcli register-finality-provider我们通过or命令在 Babylon 中注册已创建的最终性提供商fpcli rfp。

输出包含 Babylon 最终性提供商注册交易的哈希值。

```
fpcli register-finality-provider --btc-pk d0fc4db48643fbb4339dc4bbf15f272411716b0d60f18bdfeb3861544bf5ef63
{
  "tx_hash": "800AE5BBDADE974C5FA5BD44336C7F1A952FAB9F5F9B43F7D4850BA449319BAA"
}
```

当最终性提供者在 Babylon 中成功注册后，最终性提供者实例将会被启动并开始运行。

fpcli list-finality-providers我们可以通过or命令查看所有正在运行的 finality 提供程序的状态fpcli ls。该status字段可以接收以下值：

- CREATED：最终确定性提供者已创建但尚未注册
- REGISTERED：最终确定性提供商已注册，但尚未收到任何有效委托
- ACTIVE：最终性提供者拥有活跃的委托，并有权发送最终确定性签名
- INACTIVE：最终性提供者曾经处于活跃状态，但投票权已降至零
- SLASHED：由于恶意行为，最终确定性提供商被削减

```
fpcli list-finality-providers
{

    "finality-providers": [
        ...
        {
            "babylon_pk_hex": "02face5996b2792114677604ec9dfad4fe66eeace3df92dab834754add5bdd7077",
            "btc_pk_hex": "d0fc4db48643fbb4339dc4bbf15f272411716b0d60f18bdfeb3861544bf5ef63",
            "description": {"moniker": "my-name"},
            "last_vote_height": 1
            "status": "REGISTERED"
        }
    ]
}
```