去中心化数学证明协议

Mathcoin

用零知识证明求解数学开放问题

中文白皮书

Chinese Whitepaper

版本 v0.1.0（草案）

2026 年 8 月

Contents

1 摘要

Mathcoin 是一个面向数学开放问题（Open Problems）的去中心化协议。它把两个通常互不相干的领域连接起来：

- **形式化数学**：证明者使用 Lean 4 语言与 `mathlib` 数学库，将数学定理与证明严格形式化；
- **零知识证明与链上验证**：通过 Ix 平台与 Zisk 零知识虚拟机，将“一个形式化证明通过了类型检查”这一事实压缩为零知识证明，并在 BNB Chain 上由智能合约完成验证。

验证通过后，证明者自动获得 MATH 代币奖励。与此同时，MATH 通过 Flap 平台在 BNB Chain 上孵化发行：持有至少 $10{,}000$ 枚 MATH 的地址，有权按持份比例瓜分 MATH 交易额 1% 税收中 60% 分红池对应的 QQQB（Invesco QQQ Trust 代币化股票，即“纳斯达克指数代币”）分红。

> **一句话总结：**证明数学开放问题，向链上提交零知识证明，验证通过后获得 MATH；持有 $\geq 10{,}000$ MATH，持续瓜分交易额 1% 中 60% 分红池等值的 QQQB 分红。

1.1 核心价值

1. **把数学荣誉变成链上资产**。一个开放问题被正式求解后，不再只停留在期刊论文与学术会议上，而是通过密码学证明在链上锚定，成为可验证、可追溯、可激励的链上事件。
2. **证明可验证但不必须公开**。零知识证明允许证明者在不公开完整证明细节的情况下，让链上验证者相信“存在一个正确的形式化证明”。这既保护了证明者的优先权，也避免未正式发表的论文被抄袭。
3. **验证成本极低**。无论原始证明有多长、依赖多少引理，链上验证只需对约一千字节级别的证明进行一次 SNARK 验证，费用在可接受范围内。
4. **可持续的激励机制**。MATH 通过 Flap 税收与分红模型持续捕获交易价值，并以 QQQB 分红的形式回馈长期持有者，形成“证明悬赏—交易激励—长期持有”的经济循环。

2 背景与问题

2.1 数学开放问题

数学开放问题是数学共同体公认尚未被证明或否证的命题。最著名的例子包括千禧年大奖难题（Millennium Prize Problems）：黎曼猜想（Riemann Hypothesis）、P vs NP、贝赫—斯温纳顿-戴尔猜想（BSD）、纳维—斯托克斯方程的存在性与光滑性、杨—米尔斯规范场的质量间隙、霍奇猜想，以及已经解决的庞加莱猜想。

开放问题的求解往往需要数十年乃至数百年的积累。传统的求解与验证流程存在三个痛点：

- **验证周期长、成本高**。证明通常篇幅巨大、跨多个分支，期刊审稿周期以年计，错误发现滞后；
- **激励不匹配**。求解者获得荣誉，但缺乏直接的、可编程的经济激励；
- **信任中心化**。证明是否正确依赖少数专家与机构的判断，普通人无法独立验证。

2.2 形式化证明与 Lean 4

形式化证明把数学证明写成一棵可被程序检查的证明树。Lean 4 是微软研究院开发的高阶定理证明器，其 `mathlib` 库是当前最大、最活跃的数学形式化库之一，包含约两百万行形式化数学代码。

在 Lean 4 中，一个定理的证明由类型检查器（kernel）独立核验。若类型检查通过，则证明正确；若使用 `sorry`（表示“此处略去证明”）或自定义公理（`axiom`），则证明并不完整。Mathcoin 在验证流程中显式检测这些作弊路径（见第 4.3 节）。

2.3 zkPCC：零知识证明携带代码

Ix 平台提出了 **zkPCC**（zero-knowledge proof-carrying code，零知识证明携带代码）范式：将 Lean 4 程序编译为内容寻址的对象表示（ixon），并在零知识虚拟机中执行类型检查，生成可验证的零知识证明。

> **zkPCC 的核心性质：**验证者只需验证一个约 1 KB 的密码学证明，即可相信任意规模的 Lean 4 程序（乃至整个 `mathlib` 库）的类型检查正确性，且无需看到原始证明内容。

这意味着：数学证明可以成为链上经济活动的“证明资产”——它可以被验证、被交易、被悬赏，而无需公开细节。

3 Mathcoin 项目总览

3.1 使命与愿景

Mathcoin 的使命是：**用密码学与链上激励加速人类对数学开放问题的求解**。

愿景：建成全球最大的数学证明市场——任何人都可以发布问题、追加赏金、提交证明、验证证明并获得奖励；任何长期持有者都能从生态交易活动中获得真实资产（QQQB）分红。

3.2 核心参与者

角色	行为
问题发布者	锁定 MATH 赏金，发布一个形式化数学命题，声明其为开放问题。
证明者	用 Lean 4 形式化证明该命题，通过 Ix/Zisk 生成零知识证明并提交上链。
验证合约	在链上验证零知识证明，检查命题哈希、环境哈希与执行结果。
持有者	持有 $\geq 10{,}000$ MATH，按份领取 QQQB 分红，并可参与治理。
基金会	初期发布官方问题、审核问题表述、维护基础设施、申请 Binance 现货与合规。

3.3 系统组件

- **Problem Registry（问题登记合约）**：存储问题陈述的规范化哈希、赏金、状态与发布者；
- **Proof Verifier（证明验证合约）**：集成 Zisk 的 Solidity 验证器，验证 PLONK 证明并解码公共值；
- **Reward Vault（悬赏金库）**：锁定悬赏 MATH，验证通过后释放给证明者；
- **Flap Tax Vault（税收金库）**：接收 MATH 交易税收，按 FlapV3 规范处理分红与生态资金；

- **Dividend Contract（分红合约）**：由 Flap 自动部署，追踪 $\geq 10{,}000$ MATH 持有者的份额，并分发 QQQB；
- **前端（vinext）**：证明提交、问题浏览、分红领取与治理界面。

4 技术架构与证明流程

4.1 技术栈

Mathcoin 不自行发明新的 ZKP 底层，而是组合经过验证的组件：

层次	组件
形式化语言	Lean 4，`mathlib` 数学库
编译平台	Ix（Argument Computer），将 Lean 4 编译为 ixon 对象（`.ixe`）
零知识执行	Zisk（fork 自 Polygon Hermez），在 Zisk VM 中执行 ixon 类型检查
证明系统	STARK 证明 $\rightarrow$ PLONK SNARK 包装 $\rightarrow$ Solidity 验证器
链上验证	`ZiskVerifier.verifySnarkProof(bytes32,bytes32,bytes,bytes)`
发行与分红	Flap（Flap.sh，BNB Chain 上的可编程发币平台）
分红资产	QQQB：Invesco QQQ Trust (bStocks Tokenized Stock)，BNB Chain 合约 `0x205812cdbed920aff76c6580abd681a46d11efc7`

4.2 从证明到验证的完整流程

4.2.1 第 1 步：形式化命题与证明

问题发布者（或基金会）将开放问题表述为 Lean 4 定理。例如黎曼猜想的某个等价形式、哥德巴赫猜想的 Lean 4 命题等。证明者完成该命题的 Lean 4 证明，且不允许使用 `sorry` 或自定义公理。

4.2.2 第 2 步：编译为 ixon 对象

证明者使用 Ix 编译器将目标定理编译为 `.ixe` 文件：

```
ix compile Riemann.lean --consts riemannMainTheorem --out riemann.ixe
```

编译器输出内容寻址的 ixon 对象，其中包含类型检查所需的完整程序图。

4.2.3 第 3 步：Rust 内核独立类型检查

为保证不信任 Lean 4 自身工具链，使用 Rust 内核对 `.ixe` 做独立类型检查：

```
ix check-rs riemann.ixe
# 预期输出: N/N passed
```

4.2.4 第 4 步：在 Zisk VM 中执行

在 Zisk 虚拟机中执行类型检查程序，得到执行结果与资源消耗：

```
zisk-host --emulator --execute --ixe riemann.ixe
# 预期输出: failures=0, EXIT:0
```

4.2.5 第 5 步：生成 STARK 证明

执行成功后，生成 STARK 证明：

```
zisk-host --emulator --ixe riemann.ixe \\
    --proof-out riemann.proof.bin
```

4.2.6 第 6 步：包装为 PLONK 证明

为在链上高效验证，将 STARK 证明包装为 PLONK SNARK 证明：

```
cargo-zisk-dev wrap-proof --plonk \
  --proof riemann.proof.bin \
  --output riemann.plonk.bin
```

4.2.7 第 7 步：导出 Solidity 验证参数

```
cargo-zisk-dev export-solidity-calldata \
  --proof riemann.plonk.bin \
  --output riemann.fixture.json
```

得到四个验证参数：`programVK`、`rootCVadcopFinal`、`publicValues`、`proofBytes`。

4.2.8 第 8 步：链上验证与奖励释放

证明者调用 Mathcoin 的 Proof Verifier 合约，合约内部：

1. 校验 `programVK` 与协议固定的 `envHash`（防止换用其他执行环境）；
2. 调用 `ZiskVerifier.verifySnarkProof` 验证 PLONK 证明；
3. 解码 `publicValues`，检查 `failures == 0`；
4. 检查命题哈希与 Problem Registry 中登记的 `statementHash` 一致；
5. 验证通过后，从 Reward Vault 向证明者释放悬赏 MATH。

4.3 防止“平凡证明”与伪证明

4.3.1 问题一：证明了一个平凡命题

攻击方式：提交者不解决开放问题，而是证明一个显然为真的平凡定理（例如 $1+1=2$ 或 $n=n$），试图冒领赏金。

防御：Problem Registry 中登记的是目标命题的 **规范化形式化语句哈希**（`statementHash`）。链上验证时，合约要求 `publicValues` 中携带的命题哈希与登记哈希严格相等。平凡命题与开放问题的形式化语句不同，哈希不匹配，交易回滚。

4.3.2 问题二：证明了一个更弱的命题

攻击方式：目标是费马大定理（FLT）的完整命题，提交者只证明了某个具体数值情形（如 $2^3+3^3\neq 4^3$），但声称“解决了 FLT”。

防御：同样由 `statementHash` 与链上命题严格绑定。验证合约只认“与登记命题完全一致”的证明；任何弱化、强化或等价的非规范表述都需要先通过治理程序更新链上命题。

4.3.3 问题三：用 `sorry` 或自定义公理“证明”

攻击方式：在 Lean 4 中使用 `sorry` 跳过证明，或声明自定义 `axiom` 直接得到结论。

防御：协议在生成证明前，对用户提交的 Lean 4 模块运行公理与 `sorryAx` 扫描（与 Mathcoin 早期验证脚本 `CheckAxioms.lean` 一致）：

- 检查证明体中是否出现 `sorryAx`；
- 检查类型是否依赖 `sorryAx`；
- 检查用户模块是否声明自定义 `axiom`；
- 仅允许 `mathlib`、`Init` 与协议白名单中的公理。

4.3.4 问题四：使用不同版本的 Lean 环境

攻击方式：使用修改过的 Lean 4 内核或不同版本的 `mathlib`，使一个无效证明“通过”类型检查。

防御：协议固定一个官方 `lean-toolchain` 与依赖清单，并将其摘要写入合约的 `envHash`。验证时要求证明对应的 `envHash` 与官方值一致。

4.4 隐私与优先权

Mathcoin 的默认验证模式是**私有提交**：证明者仅提交零知识证明，不公开 Lean 4 证明全文。链上只记录“某命题在某时间被成功证明”，不暴露证明细节。这样：

- 证明者可以优先将论文投向期刊，不必担心链上细节泄露导致抢发；
- 任何人都可以独立验证“证明存在且正确”，但不能剽窃证明内容；
- 证明者也可以选择**公开提交**，将 `.ixe` 文件与 Lean 4 源码上传至去中心化存储，换取社区声誉与额外激励。

5 问题发布与悬赏机制

5.1 官方首批问题

协议启动初期，由 Mathcoin 基金会发布第一批开放问题，每个问题对应一个链上 `statementHash` 与基础悬赏。首批问题优先选择已有成熟 Lean 4 形式化命题、或社区形式化工作活跃的著名问题，例如：

- 哥德巴赫猜想及其弱形式；
- 孪生素数猜想；
- 黎曼猜想的形式化等价命题；
- P vs NP 的形式化表述；
- BSD 猜想、Navier–Stokes 存在性与光滑性等千禧年问题。

5.2 社区开放发布

Mathcoin 是无需许可的证明市场。任何持有 MATH 的地址都可以发布问题：

1. 发布者提供问题的 Lean 4 形式化命题，或委托基金会/社区形式化小组完成表述；
2. 发布者锁定一笔 MATH 作为初始赏金（最低额度由治理决定）；
3. Problem Registry 计算 `statementHash`，若该命题尚未被登记且表述通过形式化检查，则创建问题；
4. 其他持有者可以随时向该问题追加赏金；
5. 问题发布者或治理委员会可在一定条件下关闭/更新问题。

5.3 赏金释放

当证明者提交的零知识证明通过链上验证后，Reward Vault 自动将赏金 MATH 释放给证明者地址。释放规则：

- **首次有效证明**：获得全额赏金；
- **重复提交**：若同一命题已被证明，后续提交不再获得赏金；
- **公开提交加成**：选择公开证明全文的证明者，可从生态激励池获得额外奖励；
- **争议期**：验证通过后，赏金进入短暂争议期（例如 7 天），治理可对异常提交发起仲裁。

6 代币经济学

6.1 代币概况

项目	内容
代币名称	Mathcoin
代币符号	MATH
总供应量	1,000,000,000 MATH（10 亿，固定上限）
代币精度	18 位小数
发行网络	BNB Chain（BSC）
发行方式	通过 Flap 以 Tax Token V3 形式孵化发行

6.2 初始分配

序号	用途	比例	数量（MATH）
1	**悬赏池（Proof Bounty Pool）**	20%	200,000,000
2	**生态与 Flap 孵化（流动性/税收补贴/公开提交加成）**	25%	250,000,000
3	**团队与顾问**	15%	150,000,000
4	**基金会与研发**	15%	150,000,000
5	**公募/初始流动性**	25%	250,000,000

6.3 释放计划

- **悬赏池**：线性释放，首年释放 20%，之后每年释放 10%，直至用完；未释放部分由治理金库保管；
- **生态与 Flap 孵化**：按生态活动释放，首年不超过 50%；
- **团队与顾问**：12 个月锁仓，之后 36 个月线性释放；
- **基金会与研发**：6 个月锁仓，之后 48 个月线性释放；
- **公募/初始流动性**：按公募规则与 Flap 联合曲线流动性需求释放。

6.4 代币用途

1. **发布问题**：锁定 MATH 作为开放问题的初始赏金；
2. **追加赏金**：向已有问题追加悬赏；
3. **分红资格**：持有 $\geq 10{,}000$ MATH 的地址自动进入 QQQB 分红名单，分享交易税 60% 分红池；
4. **治理**：对官方命题表述、参数变更、金库使用、上币与审计等提案投票；
5. **证明提交费**：提交证明需支付少量 MATH（防垃圾，见下）。

6.5 防垃圾与反女巫

- 提交证明需支付一次性的 MATH 提交费（初期约为悬赏金额的 0.1%，由治理调整）；无效提交不退还；
- 问题发布与追加赏金需满足最低锁定额度与最短锁定时间；
- 分红资格要求持有 $\geq 10{,}000$ MATH，且 Flap 分红合约按地址份额计算，天然抑制小额女巫地址；
- 链上验证成本由提交者承担，进一步抬高低质量提交的成本。

7 Flap 孵化与 QQQB 分红机制

7.1 为什么选择 Flap

Flap（Flap.sh）是 BNB Chain 上的可编程发币平台（The Programmable Token Launchpad）。它提供：

- **联合曲线发币**：代币在联合曲线上完成初始价格发现，毕业（graduate）后进入 DEX 交易；
- **Tax Token V3**：支持非对称买卖税、协议强制分红、任意 ERC-20 分红代币；
- **税收金库（Vault）**：税收由 TaxProcessor 自动结算并派发到金库/分红合约；
- **SwapRegistry**：自动将税收兑换为指定分红代币（例如 QQQB）；
- **分红合约**：自动追踪持份、按比例分配分红代币。

Mathcoin 选择在 Flap 上孵化，而不是自行部署发币合约，是为了复用其经过审计的税收与分红基础设施，并把工程重心放在数学证明验证协议上。

7.2 Flap 发币参数（草案）

参数	值
代币名称 / 符号	Mathcoin / MATH
代币版本	`TOKEN_TAXED_V3`（V3 税收代币）
底池币种（quoteToken）	BNB（`address(0)`，原生币）
买入税率（buyTaxRate）	100 bps（1%）
卖出税率（sellTaxRate）	100 bps（1%）
分红比例（dividendBps）	6,000 bps（税收扣除协议费后 60% 进入分红合约）
分红代币（dividendToken）	QQQB：`0x205812cdbed920aff76c6580abd681a46d11efc7`
最低分红资格（minimumShareBalance）	$10{,}000 \times 10^{18}$（即 10,000 MATH）
市场/生态比例（mktBps）	1,000 bps（10% 进入创作者资金钱包）
销毁比例（deflationBps）	0（初期不销毁）
LP 比例（lpBps）	3,000 bps（30% 注入流动性池）

7.3 QQQB：纳斯达克指数代币

QQQB 是 **Invesco QQQ Trust (bStocks Tokenized Stock)** 的符号，即景顺 QQQ 信托的代币化股票。QQQ ETF 跟踪纳斯达克 100 指数（Nasdaq-100），因此 QQQB 可视为“纳斯达克指数代币”。

关键信息：

项目	内容
代币名称	Invesco QQQ Trust (bStocks Tokenized Stock)
代币符号	QQQB
底层资产	Invesco QQQ ETF（跟踪 Nasdaq-100 指数）
BNB Chain 合约	`0x205812cdbed920aff76c6580abd681a46d11efc7`
主要交易对	Binance 现货 QQQB/USDT; PancakeSwap V3 QQQB/USDT、QQQB/WBNB
资产类别	代币化股票 / RWA

7.4 分红流程

1. 用户在 DEX（联合曲线或 PancakeSwap）上买入或卖出 MATH，按交易额缴纳 1% 税收；
2. Flap 的 TaxProcessor 定期结算税收，并按参数扣除协议费；
3. 剩余税收按参数分配：60% 进入分红路径，30% 注入流动性池，10% 进入创作者资金钱包；
4. 分红路径的资金通过 SwapRegistry 自动兑换为 QQQB，并存入 MATH 的 Flap 分红合约；
5. 分红合约按每个地址持有的 MATH 份额（仅统计 $\geq 10{,}000$ MATH 的地址）计算可提取分红；
6. 持有者调用 `withdrawDividendsFor(address)` 提取 QQQB 到自己的钱包。

7.5 “币安交易额 1%”的口径

- 按 Flap 的税收口径执行：MATH 在 BNB Chain（币安链）上的链上交易额（联合曲线与 DEX），由 Flap 买卖税各 1% 捕获，即“币安交易额 1%”；
- 税收扣除 Flap 协议费后，按 60% 分红 / 30% 流动性池 / 10% 创作者资金钱包分配；

- 持有者最终收到的分红资产为 QQQB，由 Flap 分红合约按持份自动计算。

  持有 $\geq 10{,}000$ MATH，即可持续按份领取 QQQB 分红。

7.6 Flap 协议成本

Flap 协议对代币生命周期收取协议级费用（以官方文档为准）：

- **联合曲线阶段**：每笔买入/卖出收取交易额的 1%；
- **税收代币协议费**：最高为交易额的 0.3%，用于支撑税收结算与派发基础设施；
- **其他**：毕业到 DEX 后的池子手续费、Gas 费等由用户承担。

因此，“1% 交易额分红”的精确含义是：**税前口径为交易额的 1%；扣除 Flap 协议费（最高 0.3%）后，60% 进入 QQQB 分红池，对应净额约为交易额的 0.42%–0.6%，具体以链上参数与协议实现为准**。

8 路线图

阶段	里程碑
2026 Q3	完成白皮书；启动 vinext 前端开发；在 BNB Testnet 部署 ZiskVerifier 与 Problem Registry；用 Ix/Zisk 跑通首个形式化定理的链上验证。
2026 Q4	正式网部署验证合约与悬赏金库；通过 Flap 在 BNB Chain 孵化发行 MATH；发布首批官方开放问题（含形式化命题与哈希）。
2027 Q1	开放社区问题发布与追加赏金；上线分红领取页面；完成 Flap Vault 的第三方审计；启动 DAO 治理框架。
2027 Q2	扩大官方问题集；推动首个社区问题的有效证明；申请 Binance 现货，扩大 MATH 交易量与生态影响力。
2027 H2	生态扩展：证明市场 SDK、公开提交浏览器、与其他形式化证明社区（mathlib/Lean 社区）合作。

9 团队与治理

9.1 团队

Mathcoin 由去中心化基金会与社区开发者共同推进。核心贡献者包括（正式版补充姓名、背景与照片）：

- 形式化证明与 Lean 开发者；
- 零知识证明与 Rust 工程师；
- Solidity/Foundry 合约工程师；
- 数学顾问委员会（数论、代数几何、偏微分方程等领域专家）；
- 合规与市场顾问。

9.2 治理

Mathcoin 采用渐进式去中心化：

- 初期由基金会维护协议参数、问题表述审核与安全升级；
- MATH 持有者按持币量投票治理参数（税率、最低分红资格、最低悬赏、提交费等）；
- 关键权限（升级、紧急暂停）采用多签 + 时间锁，并逐步迁移至 DAO。

10 风险提示

1. **协议风险**：Mathcoin 依赖 Ix、Zisk、Flap 等第三方协议。这些协议可能存在未发现的漏洞、升级风险或停机风险。
2. **验证器风险**：链上 PLONK 验证器的正确性、可信设置（trusted setup）与证明系统的密码学假设可能被攻破。
3. **形式化风险**：形式化命题可能与自然语言数学命题存在偏差；命题表述错误可能导致“证明成立但未真正解决开放问题”。
4. **市场风险**：MATH 与 QQQB 价格波动剧烈；QQQB 作为代币化股票受美股市场、监管与链上流动性影响。
5. **合规风险**：代币化股票、分红机制与悬赏经济可能受到不同司法辖区的证券法、税法约束。
6. **流动性风险**：DEX 交易对可能缺乏流动性，税收与分红可能不足以覆盖成本。
7. **治理风险**：DAO 治理可能被大持币者控制，参数变更可能不利于中小持有者。
8. **智能合约风险**：合约可能存在漏洞，导致资金损失或功能失效；审计不能完全消除风险。

11 免责声明

本文档由 Mathcoin 社区与基金会编写，仅用于向公众介绍项目设计，不构成：

- 任何司法辖区内的证券发行、金融工具或投资产品要约；
- 对 MATH 或 QQQB 未来价格、收益或流动性的承诺；
- 法律、税务、投资或财务建议。

参与 Mathcoin 协议前，您应充分理解数学形式化、零知识证明、去中心化金融与代币化股票的风险，并在必要时咨询专业顾问。

12 附录：关键地址与参数

12.1 BNB Chain Mainnet 关键地址

合约/资产	地址
Flap Portal	0xe2cE6ab80874Fa9Fa2aAE65D277Dd6B8e65C9De0
Flap VaultPortal	0x90497450f2a706f1951b5bdda52B4E5d16f34C06
Flap Trigger Service	0xcf4EE25035CF883895110f367F5BA8172416a7F9
Flap Candy Box（Preview）	0x6255fbd731272a517022e99f6CaCf6A5De9414Ee
QQQB（Invesco QQQ Trust bStocks）	0x205812cdbed920aff76c6580abd681a46d11efc7
USDT（BSC）	0x55d398326f99059ff775485246999027b3197955
WBNB（BSC）	0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173BC095c

12.2 Flap 发币关键参数

参数	值
tokenVersion	`TOKEN_TAXED_V3`（枚举值 6）
buyTaxRate	100 bps
sellTaxRate	100 bps
dividendBps	6,000 bps
dividendToken	QQQB 合约地址
mktBps	1,000 bps
lpBps	3,000 bps
deflationBps	0 bps
minimumShareBalance	$10{,}000 \times 10^{18}$
quoteToken	`address(0)`（BNB）

12.3 证明流程关键命令

```
# 1. 编译 Lean 定理为 .ixe
ix compile Solution.lean --consts mainTheorem --out solution.ixe

# 2. Rust 内核独立类型检查
ix check-rs solution.ixe

# 3. Zisk VM 执行（不生成证明）
zisk-host --emulator --execute --ixe solution.ixe

# 4. 生成 STARK 证明
zisk-host --emulator --ixe solution.ixe \\
    --proof-out solution.proof.bin

# 5. 包装为 PLONK 证明
cargo-zisk-dev wrap-proof --plonk \\
    --proof solution.proof.bin \\
    --output solution.plonk.bin

# 6. 导出 Solidity calldata
cargo-zisk-dev export-solidity-calldata \\
    --proof solution.plonk.bin \\
    --output solution.fixture.json

# 7. 链上验证（Solidity）
# ZiskVerifier.verifySnarkProof(programVK, rootCVadcopFinal,
#     publicValues, proofBytes)
```