arXiv:2603.00318v1 [cs.CR] 27 Feb 2026

AESP：面向人工智能代理、具备隐私保护结算的人类主权经济协议

Jian Sheng Wang
Yeah LLC
jason@yeah.app

摘要——**随着人工智能代理日益代表人类执行经济任务，智能体自主性与人类对金融资产的控制之间出现了根本性张力。我们提出代理经济主权协议（AESP），这是一种分层协议：代理可在原生加密基础设施上以机器速度自主交易，同时在密码学上始终受限于人类定义的治理边界。AESP通过五项机制强制执行如下不变量：**代理具备经济能力，但绝不拥有经济主权：**（1）带分级升级的确定性八项检查策略引擎；（2）包含自动、显式与生物识别层级的人在回路审核；（3）带托管的EIP-712双重签名承诺；（4）基于HKDF的上下文隔离隐私与批量归集；（5）基于ACE-GF的密码学底层。我们形式化了关于安全覆盖率与延迟开销的两项可检验假设，并给出包含基线与消融设计的完整评估方法。该协议以开源TypeScript SDK实现（208项测试，十个模块），并通过MCP与A2A实现互操作性。**[1]

索引术语——**人工智能代理、代理经济、经济主权、策略执行、人在回路、EIP-712、承诺协议、MCP、A2A、隐私、临时地址、HKDF、后量子密码学、WebAssembly**

I. 引言

A. 动机

大语言模型（LLM）驱动的代理迅速部署，催生了一类新的软件：它们能够在极少人类指导的情况下进行推理、规划并执行多步骤任务[6]。当这些任务涉及金融交易——购买商品、订阅服务、结算托管——时，两项要求便会发生冲突。一方面，代理必须足够自主，才能在没有人类持续微观管理的情况下行动；若由人类批准每一次 API 调用，便违背了委托的初衷。另一方面，人类必须对资金何时以及如何流动保留主权；一个能够单方面掏空钱包的代理是不可接受的责任风险。

现有的钱包和去中心化金融界面是为人类直接交互而设计的。当前生态系统中，能够强制执行由人类定义的经济边界、面向代理的协议基本缺位。近期对区块链上自主智能体的调查 [2] 指出，信任边界是具有高价值的攻击面；而新兴的“代理经济”文献 [3]提出了代理间商务的架构，但往往假设代理拥有完全自主性，而非由人类主权掌控控制。去中心化人工智能系统中，自主性与问责之间的张力已被认定为一项根本性挑战 [4], [8]。

新兴共识指出了人工智能参与经济的两种范式：在无需许可的支付通道上持有自有钱包的完全自主代理，以及严格作为特定人类主体在法律上可问责之延伸而运作的人类委托代理。我们主张并实现了第三条路径：**主权代理经济**，即代理在加密原生基础设施上以机器速度自主进行交易，同时在密码学上仍受限于人类定义的治理边界。这将自主性与问责之间的张力从二元选择重构为一个具有可调参数的工程设计空间。

最近，来自 Google Deep-Mind 的 Tomašev 等人 [28]为“智能 AI 委托”提出了全面的概念框架，并确定了安全委托的九个维度，包括权限处理、可验证的任务完成、信任与声誉以及监控。他们的研究为该领域提供了宝贵的分类体系和设计词汇；作者指出，如何实现这些原则的具体方案仍是一项尚待解决的挑战。

缺口： 现有框架 [7], [8], [28] 明确了安全经济委托的要求，却未提供实现方案或实证验证。此前没有研究量化代理经济系统中主权执行与交易效率之间的权衡。具体而言，尚无现有系统回答以下问题：对每一项代理经济操作执行人类主权，代价是什么——会带来怎样的延迟、假阳性率和可用性影响？

假设： AESP 填补了这一缺口。我们将主张形式化为两个可检验的假设：

- **H1（安全性）：** AESP 的八项检查策略门能够自动拦截绝大多数未经授权的交易，同时将针对合法请求的假阳性率控制在个位数百分比范围内。升级单独作为运维负担（审核工作量）评估，而不计作算法安全方面的成功。

- **H2（效率）：** AESP 引入的端到端延迟开销——包括策略评估、密码学签名（通过 Rust模块）和 HKDF 密钥派生——每笔交易仍在数百毫秒的量级，与无约束基线相比，不会实质性降低交易完成率。

[1] https://github.com/ya-xyz/aesp

AESP 提供了一个具体且已实现的协议层，连接人类的数字主权实体（DSE）——即人类及其设备——与结算层（金库、托管、授权额度）。每项代理操作要么由策略许可，要么升级至人类审批。该协议借鉴了成熟的访问控制模型 [9]–[11],密码学承诺方案 [18], [19], 和区块链隐私技术 [14], [15], [17]，以提供一个经过形式化规范且可实现的人类主权代理经济框架。我们还制定了一套完整的评估方法，包括基线和消融设计（第XIII节）。

B. 设计原则

AESP 的核心不变量是：

代理应具备经济能力，但绝不能拥有经济主权。

T这一不变量通过五种机制 加以落实：

1）**策略门控执行**。每项代理操作在获准执行前，都会按照确定性的顺序接受八项策略检查（单笔交易限额、时间窗口、地址允许列表、区块链允许列表、方法允许列表、首次付款审核、最低余额和预算限制）（第 VI节）。

2）**人在回路审查**。未通过任何策略检查的操作都会转交至人的设备，以获得明确批准。关键策略变更——例如提高支出限额或扩大代理的权限范围——需要进行生物识别确认，这遵循了近期人工智能安全框架所倡导的渐进式验证模型[7]（第IX节）。3）**密码学承诺**。代理间协议采用 EIP-712 [18] 类型化数据的形式，由买方代理和卖方代理共同签名，并可选择以链上托管作为保障。双重签名要求确保任何一方都无法在承诺后单方面修改条款（第VIII节）。4）**上下文隔离隐私**。每笔交易都使用由 HKDF 派生的 [19] 临时地址，其派生上下文包含代理ID、方向、序列号和交易 ID。不同上下文会生成密码学上相互独立的地址，从而防止链上观察者关联不同代理上下文中的交易（第X节）。5）**密码学执行基底**。ACE-GF 模块 [29] 支持多链签名、基于 Argon2id/HKDF 的派生、上下文隔离地址生成，以及在 WASM 和原生 FFI 目标平台上的可选 ML-DSA-44 支持。在部署假设 DA1 下，密钥材料始终保留在密码学模块边界内（第V节）。

C. 贡献

本文作出以下贡献：

- 一份**协议规范**，对代理身份模型、策略评估算法、协商有限状态机、承诺生命周期、审核升级流程和隐私派生方案进行了形式化定义（第IV–X节）。
- 一种**密码学基础**，构建于 ACE-GF 底层架构之上[29]——以 Rust 实现，提供 WASM 和原生 FFI 绑定——支持多曲线签名（Ed25519、secp256k1、ML-DSA-44）、认证加密、Argon2id 到 HKDF 密钥派生，以及通过 REV32 钱包格式实现的上下文隔离多链地址生成（第V节）。
- 一个**开源软件开发工具包实现**，采用 TypeScript 编写，包含十个模块中的 208 项测试，[2] 涵盖身份派生、策略执行、协商、承诺、审核、MCP 工具、A2A 卡片、密码学原语以及上下文隔离隐私（第XI节）。
- 一个**评估框架**，包含四种限制程度逐步提高的基线、逐项检查的消融设计、一个不可关联性安全博弈，以及关于安全覆盖率和延迟开销的两个可证伪假设（第XIII节）。
- 一种**隐私设计**，采用 HKDF 派生的临时地址、结合时间抖动与 Fisher-Yates 洗牌的批量归集，旨在抵御切实可行的链上分析，实现交易不可关联性，同时保留可审计性（第X节）。
- **互操作性**通过模型上下文协议与现有人工智能框架实现 [21]（八项工具定义），并通过 Google 的智能体间协议实现 [22]（智能体卡片生成）（第XII节）。

D. 与现有框架的关系

本工作的一个关键定位是相对于 Tomašev 等人提出的“智能 AI 委托”框架。 [28]。表 I提供了该框架所识别的概念性需求与 AESP 中相应实现机制之间的详细映射。他们对委托维度的分类体系——权限处理、监控、可验证的任务完成、信任与声誉——为 AESP 的设计提供了概念词汇。AESP 贡献了一种具体实例化：包含 8 项确定性检查的策略引擎、人在回路审核队列、EIP-712 双重签名承诺，以及密码学身份证书。我们认为这是一项互补性贡献：概念框架与已实现的协议共同推动该领域的发展，其作用超过二者各自单独的贡献。

E. 大纲

章节 II 综述相关研究。第 III 定义系统模型、架构和威胁模型。第 IV–X 介绍协议组件：身份、密码学基础、策略、协商、承诺、审核和隐私。第 XI 介绍实现。第 XII讨论互操作性。第 XIII 介绍用于验证 H1 和 H2 的评估方法与实验设计。第 XIV 通过案例研究展示协议表达能力。第 XV 探讨局限性。第 XVI作出总结。

[2]https://github.com/ya-xyz/aesp

表 I：映射 betwee n Google DeepMind 的智能 AI 委派框架 [28]与 AESP 实现

委托维度	概念要求 [28]	AESP 实现
权限处理（§4.7）	采用渐进式信任的分级权限模型	PolicyEngine：8 项检查评估，3 个审批层级（自动/审核/生物识别），8 种关键变更类型
监控（§4.5）	持续监控和日志记录	ReviewManager：带优先级的异步审核队列，到期截止时间、6种事件类型、紧急冻结
可验证完成（§4.8）	可验证任务完成机制	CommitmentBuilder：EIP-712双重签名承诺-七状态生命周期，SHA-256 承诺哈希
信任与声誉（§4.6）	信任建立与声誉模型	带有所有者 Ed25519 签名的身份凭证，分层委托（深度 ≤ 5），基于 DID 的身份标识——tity
安全（§4.9）	密码安全与密钥管理	ACE-GF 基底（第 V；WASM/C 外部函数接口/Dart 外部函数接口）：Ed25519、secp256k1、X25519、ML-DSA-44，AES-256-GCM；密钥绝不离开密码边界
智能体通信	标准化的智能体通信协议	8 个 MCP 工具 +A2A 智能体卡片（已实现，已测试）
隐私	未明确涉及	HKDF 上下文隔离的临时地址，Fisher-Yates 洗牌，±30% 时间抖动，批量整合-tion
后量子就绪性	未涉及	通过密码学模块实现的 ML-DSA-44（FIPS 204）基于格的签名密码学模块

II. 相关工作

A. 智能体经济框架

自主经济智能体这一理念早于大语言模型。Minarsch 等人 [1]提出了 Fetch.ai 自主经济智能体（AEA）框架，其中软件智能体利用基于区块链结算的去中心化开放经济框架追求经济目标。最近，Xu [3]提出了“智能体经济”的五层架构，涵盖去中心化物理基础设施（DePIN）、去中心化标识符（DID）、认知工具、ERC-4337 结算和智能体型 DAO。他们的设计假设智能体完全自主——智能体拥有独立的经济能动性。AESP 针对同一问题领域，却采取了相反立场：智能体在人类主权之下运行，而非脱离人类主权独立运行。

Alqithami [2]对区块链上的 317 项自主智能体研究进行了全面综述，提出了由五部分组成的集成模式分类体系，并将信任边界确定为主要攻击面。Hu 等人 [4] 分析了在无需信任的底层（区块链、可信执行环境（TEE））上部署基于大语言模型的代理所面临的悖论：不可篡改性带来了防篡改能力，却失去了监督机制——而这正是 AESP 的策略门控升级模型所解决的张力。

B. 智能 AI 委托

Tomašev 等人 [28] 提出了迄今为止对 AI 委托最全面的概念性论述。其框架识别了十一项任务特征，将委托定义为一种委托代理关系，并提出了安全委托的九个维度：任务规范、资源配置、工具访问、监控、信任与声誉、权限处理，可验证的任务完成、安全以及回滚。这一分类体系为推理智能体委托提供了严谨的设计词汇；作者将具体实现确定为一个重要的开放方向。AESP 建立在其概念基础之上：表 I 将每个框架维度映射到相应的 AESP 机制，说明概念要求如何能够实现为协议组件。

C. 人在回路中的 AI 控制

从人类反馈中学习的范式由 Christiano 等人 [5],确立，他们表明，借助非专家的人类偏好反馈，且反馈量不到智能体交互的 1%，即可解决复杂的强化学习任务。Ouyang 等人 [6]通过 InstructGPT 将这一方法扩展到语言模型，证明基于人类反馈的强化学习能够产生与人类偏好相一致的输出。

在系统层面，Cheng 等人 [7] 提出了一个由三大支柱（透明度、问责、可信性）构成的模型，并采用类似于分阶段自动驾驶的渐进式验证。AESP 的分级升级机制——常规交易自动批准，异常交易需要明确审核，关键策略变更需要生物识别确认——为经济交易实现了这一渐进式方法。Hu 和 Rong [8] 指出，在密码学自我托管支撑智能体自主权时会出现问责缺口；AESP 的设计通过确保人类主体始终拥有最终经济决策权，直接解决了这一问题。

D. 策略与授权模型

AESP 的策略引擎借鉴了访问控制领域的三种传统机制。基于角色的访问控制（RBAC） [9] 不是向个人而是向角色分配权限；AESP 对此加以改编，将支出授权分配给代理 范围。基于属性的访问控制（ABAC） [10]根据主体、客体、操作和环境属性评估策略；AESP 的上下文感知策略检查遵循了这一模式。基于能力的安全 [11] 会为特定权限授予不可伪造的令牌；AESP 按策略设置、限时有效的支出授权发挥着类似作用。

近期研究已开始将这些模型应用于人工智能代理。Abaev 等人 [12]提出了 AgentGuardian，它会监控代理执行轨迹，以学习合法行为并制定自适应访问控制策略。Ganie [13] 提出将 RBAC 集成到面向工业应用的基于大语言模型的代理中。AESP 的贡献是一种混合方法：由人类主体定义静态策略，并针对超出策略范围的操作进行分级升级，同时采用分类系统检测关键策略变更。

E. 区块链隐私

公有区块链上的交易隐私已得到充分研究。隐匿地址（ERC-5564） [14] 使接收者能够接收资产，而不在公开信息中关联交易。Tornado Cash 等混币协议 [15]使用零知识证明，而 CoinJoin [16] 则合并多个输入和输出。Wichtet 等人 [17]使用交易有向无环图模型，对区块链隐私概念——不可追踪性和不可关联性——进行了形式化定义。

AESP 的隐私机制有所不同，它不需要混币或零知识证明。相反，它采用 REV32 上下文隔离派生，通过 HKDF 域分离和每笔交易的上下文标签，从身份根派生链特定密钥材料 [19]。由此生成的地址在密码学上彼此独立，无需链上混币即可实现地址级不可关联性；归集模式导致的残余侧信道泄露将在第 X节讨论。

F. 密码学承诺与托管

EIP-712 [18] 定义了一种标准，用于对类型化结构化数据进行哈希和签名，并通过域分隔符防止跨应用签名重放。AESP 使用 EIP-712 实现其双重签名支付承诺。Asgaonkar 和 Kr- ishnamachari [20] 提出了双重存款托管智能合约，以实现可证明的防欺诈交付；AESP 的承诺模型与此类设计兼容，不过该协议与结算层无关。

G. 代理互操作性

模型上下文协议（MCP） [21] 通过 JSON-RPC 提供标准化接口，以便将人工智能助手连接到外部工具。Google 的代理间（A2A）协议 [22] 通过代理卡片实现跨框架代理发现。AESP 将其经济操作作为 MCP 工具公开，并生成 A2A 智能体卡片，使任何兼容框架都能发现由 AESP 中介的服务。

H. 数字主权与自主主权身份

Allen [25]阐述的自我主权身份（SSI）概念，并通过万维网联盟的去中心化标识符 [23] 和可验证凭证 [24], 实现标准化，主张个人应掌控自己的数字身份。Krul 等人 [26]对自我主权身份组件中的信任模型进行了系统梳理。Srivastava 和 Bullock [27]考察了人工智能背景下的数字主权。AESP 将自我主权身份从身份领域拓展至经济领域：正如个人应掌控自己的标识符，他们也应掌控代理如何支配自己的资金。

I. 后量子密码学

NIST 选择 ML-DSA（原名 CRYSTALS- Dilithium）作为 FIPS 204 [30] 下的主要后量子数字签名标准，推动了针对区块链系统后量子就绪性的研究。Fernandez-Carames 和 Fraga-Lamas [31] 综述了区块链的后量子方法。AESP 的密码学模块包括 ML-DSA-44 这一可选签名方案，从而支持具备前向兼容性的代理身份和交易签名。

III. 系统模型与架构

A. 分层架构

AESP 由四层堆栈组成（见图1）：



图1：AESP 四层架构。蓝色箭头（向下）表示主权和授权流；红色箭头（向上）表示升级和事件报告。

第4层（DSE）代表人类主体及其设备（手机、浏览器扩展）。DSE 持有 ACE-GF 主助记词（REV32 编码的密封熵），制定策略，并批准或拒绝升级处理的操作。所有经济主权都归属于这一层。

第3层（AESP）实现协议逻辑：代理身份派生与认证、策略评估、协商状态管理、承诺构建、审查队列管理、密码学操作（通过基于 ACE-GF 的密码学底层）以及隐私保护地址派生。AESP 不持有资金，也不执行链上交易；它生成签名授权和承诺。

第 2 层（桥接层）向外部人工智能框架公开 AESP 操作。MCP 工具定义允许任何兼容 MCP 的代理查询余额、创建授权额度或提起争议。A2A 智能体卡片公布代理能力，以供跨框架发现。

第 1 层（结算层）负责链上执行：为托管提供资金、释放付款、管理金库和授权额度。AESP 与结算层无关；任何符合规范的实现均可作为第 1 层。

B. 参与者

我们定义三类参与者：

- **人类主体** H：ACE-GF 主助记词的持有者，以及最终的经济决策权拥有者。 H 制定策略、批准升级处理的操作，并可随时冻结任何代理。
- **代理** A_i：由 H 的主密钥在索引 i 处派生的软件实体，以 id(A_i) = SHA-256(pk(A_i)) 标识，并在 H 设定的策略边界内运行。
- **交易对手智能体** A_j：由另一人类主体 H' 控制的代理，A_i 可与其协商并形成承诺。

C. 威胁模型

我们从攻击者能力、防御保证和明确的失效模式三个方面对威胁模型进行形式化描述。

1) 攻击者能力： $\mathcal{A}_1$**：遭入侵的智能体**。攻击者控制了智能体 A_i 的运行时（例如通过提示注入、供应链遭入侵或软件漏洞）。形式上， $\mathcal{A}_1$ 可以调用 A_i 可用的任何 SDK 函数，并且可以构造带有任意参数（金额、收款方、链、方法）的任意操作请求 r 。

部署假设（DA1：必须进行策略中介）。我们假设部署架构会强制要求全部智能体签名请求在到达密码学模块之前，先经过 AESP SDK 的 PolicyEngine。具体而言，签名函数不会直接暴露给代理运行时；只能通过该 SDK 的策略门控 API 访问。这是一项必须由宿主环境强制执行的架构不变量。浏览器环境的参考部署模式是在专用 Web Worker 中加载 ACE-GF WASM 模块；该 Worker 仅通过 postMessage 暴露高级操作（签名、派生_地址、加密），并在初始化时一次性传入原始凭证材料（助记词、口令），之后绝不将其返回给调用方。主线程上的代理运行时无法直接访问 WASM 线性内存或签名导出。在 Node.js 中，可通过 worker_threads 或使用进程间通信的子进程实现等效隔离；在原生 FFI 部署中，沙箱执行环境则提供相同的边界。如果 $\mathcal{A}_1$ 能够绕过 SDK 并直接调用签名函数，策略执行保证便不复存在。在 DA1 下， $\mathcal{A}_1$ 无法修改密码学二进制文件、访问 ACE-GF 主助记词材料，也无法绕过 PolicyEngine 评估路径。

$\mathcal{A}_2$**：恶意交易对手**。代理 A_j （或其主体H'）可能操纵协商结果、否认承诺，或利用托管流程中的时机漏洞。AESP 通过双重签名承诺协议（第VIII节）缓解此类风险：托管资金到账前，双方必须对同一个 EIP-712 结构进行签名。

$\mathcal{A}_3$**：链上观察者**。 被动敌手 $\mathcal{O}$ 监控全部链上交易，并试图将其关联到单一身份或金库。AESP 的上下文隔离隐私（第 X 节）确保使用不同上下文字符串的交易生成密码学独立地址。 $\mathcal{O}$在不破解 HKDF 或攻破主密钥的情况下，无法确定两个临时地址是否属于同一主体。

2) 防御边界： 我们明确阐述以下防御保证及其边界：

- **策略执行保证：** 在部署假设 DA1（强制策略中介）下，任何违反当前策略的操作都无法到达签名层，除非先被路由至 ReviewManager。这一保证是架构层面的：它依赖宿主环境确保代理运行时无法直接访问密码签名API。在未实施 DA1 的部署中（例如，代理进程可直接访问签名导出），遭入侵的智能体可以完全绕过策略门。
- **密钥隔离保证：** 私钥永远不会离开 Rust 模块的内存空间。在 WASM 部署中，签名在 WASM 线性内存中执行；在原生 FFI 部署中，密钥保留在 Rust 进程的地址空间内。这两种路径都会在对象销毁时强制清零。如果对手能够修改编译二进制，或利用宿主环境中的运行时漏洞，这一保证就会失效。
- **主权保证：** 人类主体 H 可以随时冻结任何代理，并立即停止全部经济活动。只要 DSE 设备未遭入侵，这一保证就成立。

3）失效案例：我们指出 AESP 尚未能完全缓解的三种失效模式，并对此进行坦诚讨论：

共谋攻击。如果买方代理 A_i 和卖方代理 A_j 均遭到入侵（或其主体串通共谋），他们就可以构造互利但违反策略的承诺。AESP 的策略门按代理分别运行；检测跨代理共谋需要借助声誉或异常检测层，这超出了 AESP 当前的范围。

审批疲劳。如果对手触发大量审核请求（例如，30 分钟内发出 50 个请求），人类主体可能会开始不经仔细检查就批准请求。AESP 的审核系统支持紧急级别和到期截止时间，但目前尚未实现速率限制或疲劳检测。我们在第 XIII-D 节量化这一风险。

时间窗口边界利用。 了解策略时间窗口边界的对手可以将交易排队，等到窗口开启的那一刻执行，从而可能在人工来得及反应之前压垮预算跟踪器。

AESP 的滚动预算限制（日/周/月）提供了额外的防护，但如果单笔交易限制设得过宽松，那么在单个评估周期内突发的大量交易仍可能超出预期限制。

4) 不在范围内： 我们不防御以下情况：(1) 结算层遭到入侵（这需要与 AESP 无关的链上安全保障）；(2) DSE 设备遭到入侵（如果对手控制了人类的手机，AESP 的所有保障都将失效）；(3) 对密码学模块发起侧信道攻击（这需要在硬件层采用常数时间实现）；或 (4) 通过社会工程诱使人类主体批准恶意请求（这需要开展用户教育，而非进行协议设计）。

IV. 身份

A. 代理派生

AESP 根据人类主体的 ACE-GF 助记词，以确定性方式派生代理身份。在 REV32 模式下，助记词会解码为 32 字节的 REV 载荷 r_H，并通过 ACE-GF 流程派生密钥材料：

$$\text{root}_H = \text{HKDF}(\text{Argon2id}(p_H,\ \text{salt}(r_H)),\ \text{"acegf:identity:root"}) \tag{1}$$

其中 p_H 是所有者的口令域，$\text{salt}(r_H)$从 REV32 载荷中提取盐值部分（32 字节载荷划分为盐值和熵字段的方式由 ACE-GF 规范定义， [29]）。HKDF 符号遵循 RFC 5869 [19]；完整的提取与扩展分解见公式5 (第V-C节）。随后，在代理作用域派生上下文中，以确定性方式从root_H派生代理专用密钥，为每个代理索引 i生成$(\text{sk}_i,\ \text{pk}_i)$。

$$\text{id}(A_i) = \text{SHA-256}(\text{pk}_i) \tag{2}$$

代理被分配了一个形式为 did:aesp:id(A_i) 的去中心化标识符（DID） [23]。这一推导是确定性的：相同的 ACE-GF 助记词、口令域和代理索引会产生相同的代理身份，从而支持确定性恢复。

B. 身份凭证

为建立可验证的授权，人类主体 H 会为每个代理签发一个身份凭证：

$$\begin{aligned}\text{cert}(A_i) = \langle &\text{ver}, \text{id}(A_i), \text{pk}_i, \text{xid}_H, C, h(\pi),\\ &\alpha_{\max}, \text{chains}, t_{\text{created}}, t_{\text{expires}}, \sigma_H\rangle\end{aligned} \tag{3}$$

其中，ver 是凭证版本（当前为“1.0”），xid_H 是人类的 xidentity 公钥， C 是授予的能力集合（例如，{payment, negotiation}）， $h(\pi)$ 是相关策略对象的 SHA-256 哈希值， $\alpha_{\max}$ 是自主交易金额上限， σ_H 是人类针对凭证字段生成的 Ed25519 签名。

验证需要受信任所有者 xidentity xid_H。验证者会检查 σ_H 是否为由 xid_H对规范序列化内容生成的有效 Ed25519 签名，证书是否尚未过期，以及所声明的能力是否在相关策略的范围内。

C. 代理层级

AESP 支持分层委托，最大深度可配置为 $d_{\max} = 5$。该层级构成一棵以人类主体为根节点的有根树：

$$H \to A_0 \to A_{0,1} \to A_{0,1,2} \to \cdots \tag{4}$$

每个代理只能委派自身能力的子集。AgentHierarchyManager 强制执行深度约束，并提供一个升级链函数：给定代理 A_i，返回用于审核路由的有序序列$[A_i$, 父代理（A_i）. .. ， $H]$。如果代理无法解决策略违规问题，审核请求将沿该链向上升级，直至人类主体。

V. 密码学基础

AESP 使用 ACE-GF 作为其密码学执行层 [29]。该层提供确定性身份重建、上下文隔离派生和多曲线签名能力，这些都是 AESP 所需的。相同的核心实现
可通过 WebAssembly 和原生 FFI 目标使用，
从而确保在浏览器、
服务器和移动端部署中具有一致的密码学行为。

A. 架构

ACE-GF 基底采用分层实现：

- **核心派生层**：REV32 重构、HKDF 域分离，以及按上下文隔离的链专属密钥流。
- **密码运算层**：提供签名、加密和密钥协商 API，并支持特定曲线分派。
- **签名器层**：为以太坊虚拟机、Solana、比特币及其他受支持的生态系统提供特定链交易签名。
- **平台绑定**：WASM 和原生 FFI 接口，在适配宿主运行时的同时保持单一密码学核心。
- **安全工具**：基于 Argon2id 的凭证加固、REV32工件处理和内存清零。

B. 支持的密码学原语

表 II 总结了密码学模块中提供的密码学原语（除非另有说明，全部原语均适用于全部平台目标）。

C. *REV32*钱包格式与上下文隔离派生

密码层的核心创新是**REV32钱包格式**，它采用基于身份根的上下文隔离密钥生成（而非HD路径扩展）。派生流水线的运作方式如下：

1) **REV32恢复**。 将ACE-GF助记词解码为32字节的REV32载荷（密封熵 + 元数据）。 2) **身份根派生**。 从密封REV32秘密和口令域派生身份根，然后从该根派生链特定密钥材料。

表 II：ACE-GF 基底中的密码学原语

类别	原语	在 AESP 中的用途
签名	Ed25519	代理身份，Solana 交易
	ECDSA/secp256k1	以太坊虚拟机交易，EIP-712 签名
	Schnorr/secp256k1	比特币 Taproot 交易
后量子	ML-DSA-44（FIPS 204）	面向未来的代理证书
密钥协商	X25519 ECDH	端到端加密协商消息
加密	AES-256-GCM	协商消息加密
	AES-256-GCM-SIV	口令封装
	ChaCha20-Poly1305	替代加密
KDF	Argon2id	口令 + REV32 →
	HKDF-SHA256	上下文隔离派生
Hash	SHA-256	身份、承诺

3) **上下文扩展**。 给定一个上下文字符串 ctx，使用 HKDF 计算派生密钥：

$$\begin{aligned} \text{prk} &= \text{HKDF-Extract}(\text{salt}, \text{identity_root}) \\ \text{dk} &= \text{HKDF-Expand}(\text{prk}, \text{info}, L) \end{aligned} \tag{5}$$

其中 info 为 = "ACEGF-REV32-V1-"‖curve‖":"‖ctx， L 是目标曲线的密钥长度。

4)**地址生成**。 派生密钥 dk 用于计算链特定地址（例如 Solana 的 Ed25519 公钥、以太坊虚拟机的 ECDSA 公钥的 Keccak-256 哈希等）。

该流程支持七种地址命名空间：Solana、以太坊/以太坊虚拟机和比特币（包括 Taproot）具有完整的签名器实现；Cosmos 和波卡支持地址派生；ML-DSA-44 提供后量子命名空间；xidentity 则作为 AESP 的原生身份命名空间。

关键属性是 **上下文隔离**：给定两个不同的上下文字符串 $\text{ctx}_1 \neq \text{ctx}_2$，派生密钥$\text{dk}_1$ 和 dk_2在计算上相互独立。这一属性源于 HKDF 的伪随机性[19], 是 AESP 隐私机制的基础（第 X节）。

密码学模块通过统一 API 提供上下文隔离的派生功能：

$$\texttt{view_wallet_unified_with_context}(\text{credential}, \text{ctx}) \tag{6}$$

其中， $\mathcal{W}$ 包含在给定上下文下为全部支持的链派生的地址。TypeScript 软件开发工具包调用此函数以生成保护隐私的临时地址。

D. 密钥保护与内存安全

密码学模块实现了多项密钥保护措施：

- **内存隔离**。 全部密码学操作均在 Rust 模块自身的内存空间中执行——在浏览器/Node.js 部署中使用 WASM 线性内存，在 C 外部函数接口和 Dart FFI 部署中则使用本机进程堆。无论哪种情况，宿主运行时只能看到公钥、地址和签名输出；私钥绝不会越过绑定边界。
- **清零**。 所有敏感数据结构都实现了 Rust 的 Zeroize 和 ZeroizeOnDrop 特征，确保密钥材料离开作用域时会被覆盖。
- **Argon2id 封装**。 需要持久化钱包秘密材料时（例如存储在浏览器存储或应用密钥链中），会使用 AES-256-GCM-SIV 对其进行保护，所用密钥通过 Argon2id 从用户口令派生。默认参数（内存成本 = 4 MB、时间成本 = 3、并行度 = 1）针对资源受限的 WASM 环境进行了调整；原生 FFI 部署可以提高这些参数，以符合 OWASP 建议。
- **不导出密钥**。WASM 和 C 外部函数接口绑定都不会暴露原始私钥字节。sign_* 和 derive_* 函数接受钱包凭证和上下文，在内部执行操作，并且只返回结果。

E. 后量子就绪性

密码学模块包含 ML-DSA-44（模块格数字签名算法，FIPS 204 [30]），这是 NIST 选定的后量子签名标准。ML-DSA-44 签名可作为一种可选签名方案，用于代理身份证书和交易授权。这为需要抵御量子计算攻击的 AESP 部署提供了迁移路径，同时不会影响采用 Ed25519 或 secp256k1 的现有经典安全部署。

F. 支持上下文的 *EIP-712* 类型化数据签名

密码学模块原生支持带上下文支持的 EIP-712 [18] 类型化数据签名：

$$\text{evm_sign_typed_data_with_context}(\text{凭证}, \text{上下文}, \text{typed_da} \tag{7}$$

该函数在给定上下文下派生 secp256k1 密钥对（使用 REV32 流水线），然后对类型化数据的 EIP-712 哈希进行签名。这使得 AESP 的承诺协议（第 VIII）能够使用上下文隔离的密钥来签署承诺，从而提供额外一层隐私保护：某项承诺的签名密钥由该交易特有的上下文字符串派生，$\mathcal{W}$从而防止基于签名的关联。

VI. 策略引擎

A. 策略结构

A policy π is a tuple:

$$\pi = \langle \text{id}, \text{agentId}, \text{xid}_H, s, \kappa, t_{\text{created}}, t_{\text{expires}} \rangle \tag{8}$$

其中 $s \in$ {自动_支付，协商，承诺，完整}为策略范围，且 κ 为条件记录，包含表 III中所示的字段。

各范围按优先级排序，以启用升级检测：rank(auto_支付) = 1、 rank(协商) = 2、 rank(承诺) = 3、 rank(全部) = 10。

表 III：策略条件字段

条件	Type	语义
每笔交易最大金额	数字	每笔交易最大金额 s
每日最大金额	数字	滚动 24 小时限额
每周最大金额	数字	滚动 7 天限额
每月最高金额	数字	日历月限额
允许列表地址	字符串[]	允许的接收方
允许列表链	字符串[]	允许的链
允许列表方法	字符串[]	允许的方法
时间窗口	{start, end}	营业时间（HH:MM）
最低交易后余额	数字	最低交易后余额
首次付款需审核	布尔值	首次付款需要审核 w

B. 策略评估算法

当代理 A_i 请求操作 r时，策略引擎按顺序评估 A_i 的全部活动策略。对于每项策略π，引擎按固定顺序执行八项检查：1)**单笔交易金额**。如果r.金额>κ.maxAmountPerTx，则拒绝。2) **时间窗口**。如果定义了 κ.timeWindow，且当前时间 t 不在 [start、end] 范围内（处理跨午夜情况），则拒绝。3) **地址允许列表**。如果 κ.allowListAddresses不为空，且 r.to不在 $\notin \kappa$.allowListAddresses中，则拒绝。4) **区块链允许列表**。如果 κ.allowListChains不为空，且r.chain不在 $\notin \kappa$.allowListChains中，则拒绝。5) **方法允许列表**。如果 κ.allowListMethods不为空，且 r.method不在 $\notin \kappa$.allowListMethods中，则拒绝。6)**首次付款审核**。如果 κ.requireReviewFirstPay为 =true，且 A_i 在 π下没有此前成功的支付，则拒绝。7) **最低余额**。如果 κ.minBalanceAfter已定义，且预计交易后余额 $< \kappa$.低于 > 0 minBalanceAfter，则拒绝。8) **预算限制**。查询预算跟踪器中 A_i的滚动每日、每周和每月总额。如果加入 r.金额会超出任一限制，则拒绝。

如果某项策略的全部八项检查均通过，则该操作将被自动批准，并返回匹配的策略ID。如果没有策略允许该操作，则将其转交至审核子系统（第IX节）。

析取（OR）语义。 当代理有多个有效策略时，评估遵循析取 （OR）语义：如果 任一 匹配的策略允许该操作，则该操作获批准（算法 1：循环在第一个通过的策略处返回）。这是一种有意采用的基于能力的设计 [11]——每项策略授予一项特定能力（例如“向供应商 X 支付最多 100 USDC”），拥有多个策略的代理将获得这些权限的并集。使用场景包括针对各供应商的支出限额、具有不同阈值且共存的日常运营策略与紧急策略，以及与永久基础策略重叠的限时临时策略。OR 语义的一个后果是，权限最宽松的 匹配策略决定结果；人类主体有责任避免签发范围过宽的策略。这一责任通过关键策略变更分类（表 IV），对于任何扩大代理权限的变更，均要求人工授权，最高要求进行生物识别确认。

pend

算法 1 策略评估 要求：操作请求 r、代理 A_i、有效策略 Π_{A_i} 确保：决定 $\in \{$ 批准 (π) 、审核 _ 必需 $\}$1：对每个 $\pi \in \Pi_{A_i}$ 执行 2：$\kappa \leftarrow \pi$.条件 3：如果 r.金额 $> \kappa$.每笔交易最大金额 则 4：继续 5：结束如果 6：如果 ¬在时间窗口内(t、 κ.时间窗口) 则 7：继续 8：结束如果 9：如果 κ.允许列表地址 $\neq \emptyset \wedge r$.至 $\notin \kappa$.允许列表地址 则 10：继续 11：结束如果 12：如果 κ.允许列表链 $\neq \emptyset \wedge r$.链 $\notin \kappa$.允许列表链 则 13：继续 14：结束如果 15：如果 κ.允许列表方法 $\neq \emptyset \wedge r$.方法 $\notin \kappa$.允许列表方法 则 16：继续 17：结束如果 18：如果 κ.首次付款需审核 $\wedge$¬有先前支付(A_i、 π) 则 19：继续 20：结束如果 21：如果 预计余额(r) $< \kappa$.最低交易后余额 则 22：继续 23：结束如果 24：如果 ¬在预算内(A_i、 r.金额) 则 25：继续 26：结束如果 27：返回 已批准(π) 28：结束循环 29：返回 审核_必需

C. 关键策略变更分类

修改策略时，AESP 会对变更进行分类，以确定所需的审批级别。共识别出八种变更类型，如表 IV所示。

生物识别集合中的变更要求人类主体在其移动设备上通过生物特征认证（例如 FaceID、指纹）进行确认。审核集合中的变更要求通过审查界面明确确认。

VII. 协商

A. 状态机

AESP 中的智能体间协商被建模为一个包含八种状态和十三个状态转换的有限状态机（FSM）。

表 IV：关键策略变更分类

变更类型	条件	审批
预算_增加	任意上限*增加	生物识别
范围_升级	rank(new) > rank(old)	生物识别
addr_移除 _all	已清除允许列表	生物识别
addr_add	已添加新地址	审核
time_窗口 _移除	时间限制已移除	审核
min_余额 _较低	最低余额已减少	审核
首次_支付_禁用	首付审核关闭	审核
到期时间_延长	有效期已延长	审核

状态： $S = \{s_0, s_1, \ldots, s_7\}$ 如表 V 中所定义。

表 V：谈判有限状态机状态

状态	Name	描述
s_0	初始	会话已创建，无消息
s_1	报价 _sent	发起方已发送报价
s_2	已收到要约_	响应方已收到报价
s_3	还价中	正在进行反要约交换
s_4	已接受	双方均同意
s_5	已拒绝	协商已终止
s_6	已承诺	已附加 EIP-712 承诺
s_7	争议中	承诺作出后提出争议

状态转换：有效状态转换函数 δ： $S \times M \to S$由十三个元组定义：

$$\begin{aligned}\delta = \{&(s_0, \texttt{offer}) \to s_1,\ (s_0, \texttt{offer_recv}) \to s_2,\\ &(s_1, \texttt{counter}) \to s_3,\ (s_1, \texttt{accept}) \to s_4,\\ &(s_1, \texttt{reject}) \to s_5,\ (s_2, \texttt{counter}) \to s_3,\\ &(s_2, \texttt{accept}) \to s_4,\ (s_2, \texttt{reject}) \to s_5,\\ &(s_3, \texttt{counter}) \to s_3,\ (s_3, \texttt{accept}) \to s_4,\\ &(s_3, \texttt{reject}) \to s_5,\ (s_4, \texttt{commit}) \to s_6,\\ &(s_6, \texttt{dispute}) \to s_7\}\end{aligned} \tag{9}$$

请注意， $s_3 \to s_3$ (计数器 $\to$ 计数器)是一个自循环，允许进行多轮协商。会话受最大轮数（默认值：10）和生存时间（默认值：24 小时）的限制。当前 FSM 仅强制执行会话级生存时间；每状态超时（例如，已接受状态的承诺期限短于 24 小时）则交由应用层处理。这种设计使核心 FSM 保持简单，同时允许在此基础上叠加针对具体部署的超时策略。

B. 消息格式与加密

协商消息分为四种类型：negotiation_offer、negotiation_counter、negotiation_accept 和 negotiation_reject。在代理之间传输时，消息采用端到端加密：通过 X25519 ECDH 密钥协商（由密码学模块执行）派生共享密钥，然后使用 AES-256-GCM 进行对称加密。加密信封包含版本标识符、算法描述符、Base64 编码的密文、用于重放保护的唯一消息标识符，以及时间戳。

该协议与传输方式无关：加密消息可通过 MCP 工具调用、HTTP、WebSocket 或任何其他信道传输。NegotiationProtocol 类接受可插拔的 MessageSender 接口，以抽象传输方式。

C. 协议哈希

当双方均接受后，将计算协议哈希：

$$h_{\text{agree}} = \text{SHA-256}(\text{JSON}(\text{lastRound.payload})) \tag{10}$$

其中 `lastRound.payload` 是最近一次交换的报价或反要约。此哈希包含在接受消息中，并可绑定到 EIP-712 承诺（第 VIII节），以通过密码学方式将协商结果与付款承诺关联起来。

VIII. 承诺

A. EIP-712 结构

AESP 使用 EIP-712 [18] 类型化结构化数据来处理代理与代理之间的付款承诺。域分隔符为：

$$D = \langle \texttt{name} : \text{“YalletAgentCommitment”},\ \texttt{version} : \text{“1”},\ \texttt{chainId} : c \rangle \tag{11}$$

承诺类型包含九个字段：buyerAgent（地址）、sellerAgent（地址）、商品（string）、价格（uint256）、货币（地址）、deliveryDeadline（uint256）、仲裁人（地址）、escrowRequired（bool）以及随机数（uint256，来自 crypto.getRandomValues）。

B. 双重签名流程

承诺会经历以下生命周期：

$$\texttt{draft} \to \texttt{proposed} \to \texttt{buyer_signed} \to \texttt{fully_signed} \to \texttt{escrowed} \to \texttt{delivered} \to \texttt{completed} \tag{12}$$

并可选择转换为争议中（从已托管或已交付状态转换）或已取消（从任何托管前状态转换）。承诺哈希为：

$$h_c = \text{SHA-256}(\text{JSON}(\langle D, V \rangle)) \tag{13}$$

其中， V是承诺值记录，JSON(·) 表示使用按字典序排序的键进行确定性 JSON 序列化（与软件开发工具包的 JSON.stringify 配合排序键规范化的结果一致）。买方和卖方均基于相同的规范形式独立计算 h_c并签署；只有在双方签名都存在时，才会达到完全_签署状态。

启用上下文隔离签名后，各方会使用密码学模块的 evm_sign_typed_data_with_context 函数，并传入承诺专属上下文字符串，以确保签名密钥对本次交易是唯一的。

C. 结算集成

AESP 不负责实现托管或资金转账。它会生成一份完整签名的承诺记录，其中包含 EIP-712 结构、双方签名以及元数据（托管交易哈希、交付确认哈希、释放交易哈希）。结算层会使用此记录为托管注资、验证交付并释放支付。这种分离确保 AESP 与结算层无关。

IX. 审核（人在回路）

A. 审核队列

当策略引擎拒绝某项操作（第VI节）时，AESP会创建一个审核请求，并将其放入优先队列。每个请求包含原始操作、代理ID、策略违规原因、紧急级别（低、正常、高、严重）以及过期截止时间（默认：30分钟）。

审核子系统返回一个Promise，该Promise在人类提交响应（批准、拒绝或修改）时兑现，或在截止时间到期时拒绝。此设计允许调用代理在无需轮询的情况下等待审核结果。

B. 升级层级

AESP 定义了三个升级层级（表VI）。

表 VI：升级层级

Tier	触发器	所需操作
自动	全部策略检查均通过	无需人工参与
审核	策略检查失败	人工通过界面确认
生物识别	关键策略变更	人工生物识别认证

这种分级机制实现了渐进式验证 [7]：常规操作可顺畅进行，异常操作需要人工检查，高风险变更则要求亲临设备现场。

C. 紧急冻结

人类主体可随时通过 DSE 发出冻结命令，冻结任何代理。代理被冻结后：(1) 该代理所有待处理的审核请求都会立即被拒绝；(2) 任何新操作都会被拦截，并返回 AGENT_FROZEN 错误；(3) 冻结状态会持久保存，即使重启也不会消失。冻结是最后的手段，确保人类始终能够停止代理的经济活动。

D. 事件系统

审核系统会发出六种事件类型：请求_已创建、请求_已批准、请求_已拒绝、请求_已修改、请求_已过期，以及请求_已取消。外部系统（例如移动通知服务、审计日志记录器）可以订阅这些事件，以实时了解审核流程。

X. 隐私

A. 问题陈述

在公有区块链上，所有交易对任何观察者 $\mathcal{O}$都可见。如果代理在多笔交易中使用同一地址， $\mathcal{O}$ 就能轻易将它们关联起来。即使使用不同地址，时间模式、金额和交易对手也可能导致统计相关性。AESP 的隐私子系统旨在解决交易不可关联性问题：对于由 AESP 生成的两笔交易 tx_1 和 tx_2 ，不持有主密钥的观察者$\mathcal{O}$ 应当无法判断它们是否源自同一个金库或主体。

B. 上下文隔离地址派生

AESP 使用 HKDF [19] 和上下文字符串派生临时地址，该字符串编码了交易的身份上下文。上下文字符串通过按字母顺序对键值片段排序，并以冒号连接而构成：

$$\text{ctx} = \text{sort}(\{\texttt{agent:}a,\ \texttt{dir:}d,\ \texttt{seq:}n,\ \texttt{tx:}t\})\|\texttt{':'} \tag{14}$$

其中， a 表示代理ID， $d \in$ {表示入站，出站}， n 表示序列计数器， t 表示唯一的交易通用唯一标识符。排序可确保上下文字符串的生成具有确定性。

上下文字符串会传递至密码学模块的 REV32 派生流程（第V-C节），该流程使用 HKDF，并采用以下参数：

$$\text{info} = \texttt{"ACEGF-REV32-V1-"}\|\text{curve}\|\texttt{":"}\|\text{ctx} \tag{15}$$

由于不同的上下文字符串会产生密码学独立的 HKDF 输出，因此，在不知道主密钥的情况下，无法关联从不同上下文派生的地址。

C. 隐私级别

AESP 支持三个隐私级别：

- **透明** (ℓ_0)：代理直接使用其主金库地址。不提供隐私保护。

- **基础** (ℓ_1)：每个代理-链-方向三元组共用一个地址。上下文： {代理: a，方向:d，模式:basic}。
- **隔离** (ℓ_2)：每笔交易使用一个唯一地址。上下文包含唯一的交易 ID，确保地址仅使用一次。

D. 地址池

为避免交易时的派生延迟，AESP 会为每个代理-链-方向三元组预先派生一组临时地址。默认池大小为 5。地址被领取后，地址池会异步补充。池中地址使用上下文段 pool:pre 和递增的序列计数器；地址被领取时，会映射到特定的交易上下文。

E. 归集

临时地址会累积资金，最终必须将这些资金归集回金库。简单地在单笔交易中扫描所有地址并归集资金，会重新建立它们之间的关联。AESP 的 ConsolidationScheduler 通过以下三种技术缓解这一问题：

时间抖动。 基础合并间隔（默认：4小时）会按可配置的抖动比（默认：±30%）进行扰动：

$$t_{\text{actual}} = t_{\text{base}} \cdot (1 - \rho + 2\rho \cdot r), \quad r \sim \text{Uniform}(0, 1) \tag{16}$$

其中 ρ 为抖动比。

地址乱序。 在归集之前，临时地址集合使用Fisher-Yates算法进行置换：

$$\text{for } i = n{-}1 \text{ downto } 1: \ j \leftarrow \text{random}[0, i]; \ \text{swap}(a_i, a_j) \tag{17}$$

从而确保归集顺序不会泄露任何有关派生顺序的信息。

分批执行。 地址按可配置的批次大小（默认：5）进行归集，批次间的随机延迟从 $[10, 60]$ 分钟中均匀抽取。

F. 审计标签

尽管能够保证不可关联性，资产所有者仍必须重建交易历史以进行审计。AESP 的 ContextTagManager 会为每笔临时交易创建一个加密的上下文标签，记录代理ID、policyID、承诺ID、临时地址和元数据。标签会存储在本地，也可选择以加密形式归档到永久存储（例如 Arweave）。

归档支持三种批处理策略：即时（逐个归档每个标签）、时间窗口（在可配置的窗口期后批量归档，默认：5 分钟）以及计数阈值（计数达到阈值时批量归档，默认：50）。

G. 隐私设计原理

隐私设计基于以下论证。在地址派生层，HKDF [19] 使用不同的 info 字符串会生成在计算上与独立随机值不可区分的输出，前提是输入密钥材料具有足够的最小熵（在 REV32 模式下，REV 载荷包含 224 位熵）。这意味着，从不同上下文字符串派生的地址在密码学上彼此无关——只看到链上地址的观察者无法判断这些地址是否共享同一个主密钥，除非攻破 HKDF 的伪随机性。

然而，在实践中，地址级不可关联性并不意味着完全的交易不可关联性。即使地址彼此独立，侧信道——归集时序、交易金额、交易对手模式——仍可能泄露关联信息。AESP 的归集对策（时间抖动、Fisher-Yates 洗牌、分批执行）旨在减少这些侧信道，但我们并未声称能够抵御全部对手，实现形式化的不可关联性。第 XIII-C节介绍了计划开展的实证评估，以考察启发式聚类敌手下残余的可关联性。形式化隐私在基于仿真的框架中进行分析（例如 Wicht 等人 [17] 的交易有向无环图模型）留待未来工作。

XI. 实现

A. SDK 架构

AESP 以 TypeScript 库（`@yallet/aesp`）的形式实现，目标为 ES2022，并输出 ECMAScript 模块（ESM）。实现分为十个模块，每个模块都有独立的入口点，以支持摇树优化（表 VII）。

表 VII：AESP SDK 模块

模块	主要导出项
类型	全部共享类型定义
密码学	签名、加密、哈希
身份	deriveAgentIdentity、证书
策略	PolicyEngine、预算跟踪器
协商	协商状态机、协议
承诺	CommitmentBuilder
审核	ReviewManager
`mcp`	MCP服务器，工具定义
`a2a`	代理卡构建器
隐私	地址池管理器，归集

B. 密码学后端

所有密码学操作均由第 V 节所述的 Rust 模块执行。TypeScript SDK 通过一个精简的适配器层与该模块交互，该层负责序列化/反序列化：在浏览器/Node.js 部署中通过 `wasm-bindgen` 实现，在服务器端和移动端部署中则通过 C 外部函数接口绑定实现。仅针对 SHA-256 哈希，在密码学模块不可用时提供 Web Crypto API 备用方案。

C. 存储抽象

所有有状态模块都接受可插拔的存储适配器接口：

```
接口 StorageAdapter { getItem(key:string): Promise<string | null>; setItem(key:string, value: string): Promise<void>; }
```

这使其能够在不同环境中部署（用于测试的内存存储、浏览器的 localStorage、服务器的数据库）。

D. 测试

该实现包含 208 项测试，分布在 ninetest 个测试套件中（涵盖全部十个模块），并使用 Vitest 执行。测试覆盖身份派生、策略评估（包括全部八种检查类型和全部八种关键变更分类）、协商 FSM 状态转换、承诺生命周期、审核队列和冻结机制、MCP 工具验证、A2A 卡片生成、密码学操作以及隐私功能（地址派生、合并抖动、Fisher-Yates 洗牌、审计批处理）。

XII. 互操作性

A. 模型上下文协议（*MCP*）

AESP 定义了八种 MCP [21] 工具，可向任何兼容 MCP 的人工智能框架开放经济操作（表 VIII）。[3]

表 VIII：MCP 工具定义

Tool	描述
yault_检查_余额	查询代理账户余额
yault_存入	存入 ERC-4626 金库
yault_赎回	赎回金库份额
yault_创建_额度	创建支付额度
Yault_取消_额度	取消额度
Yault_提交_争议	提交带证据哈希的争议
Yault_查询_预算	查询剩余预算
yault_list_代理	列出子代理及其状态

每个工具都包含用于参数验证的 JSON 架构定义。MCPServer 类提供支持标准输入输出和服务器发送事件（SSE）的传输层。

B. 智能体间（*A2A*）协议

AESP 根据代理配置生成 A2A 智能体卡片 [22]。每张卡片都会公布代理的能力（支付、协商、数据查询、承诺、委托、仲裁）、支持的输入/输出模式、身份验证方案（Ed25519）和端点 URL。这使跨框架代理发现成为可能：运行在不同平台上的代理可以发现由 AESP 管理的代理的经济能力并发起交互。

XIII. 评估方法

本节规定了用于验证第I节所述两项假设的评估方法。所有实验均在SDK层执行，使用AESP TypeScript库和密码学模块（主要基准环境的WASM绑定）；无需链上交易。实验框架、运行脚本和复现说明与源代码一并维护。[4] 本版本侧重于方法和实验设计；实证结果表将留待专门的评估报告。

A. 评估设计

1）请求语料：我们定义了一个包含1,500个操作请求的合成请求语料：

- **攻击集**($n = 1,000$)：代表未授权代理活动的请求，分为两类。单条件违规($n \approx 950$) 各自至少违反一项策略条件，涵盖：超额金额、超出窗口时间、不允许地址、不允许链、不允许方法、首次支付违规、最低余额违规和超预算请求。聚合违规($n \approx 50$) 单独来看均符合策略——每项请求单独通过全部八项检查——但代表只能通过跨请求上下文检测到的未授权行为（例如，在追踪器评估窗口之间快速连续发起符合策略的交易，导致预算总额耗尽）。
- **合法集**（$n = 500$）：符合全部策略条件的请求，涵盖不同金额（0.01–100单位）、允许的地址、允许的链以及窗口内时序的正常交易模式。金额从对数正态分布中抽取，以模拟真实的消费模式。

每个请求都是一个结构化的 PolicyCheckRequest 客体，包含金额、收款地址、链ID、方法、时间戳和余额上下文等字段。参考策略采用具有代表性的条件：每笔交易最大金额 = 100，每日最大金额 = 500，每周最大金额 = 2，000，每月最高金额 = 5，000，时间窗口 = 09:00–21:00，最低交易后余额 = 10，首次付款需审核 =true，允许列表中包含 10 个地址、3 条链和 4 种方法。

2）基线：我们将 AESP 与四个限制程度逐步提高的基线进行比较。由于不存在标准化的公开基准套件，也不存在能够以相同方式公开八重检查门及分级人工升级机制的可直接比较的已部署协议，因此，这些基线被设计为机制隔离控制：每一步增加一层治理复杂度，使我们能够将安全性和延迟的变化归因于特定的协议组件，而不是无关系统之间的实现差异。

- **B0（无约束）：** 无策略检查。全部请求均自动批准。这代表一个没有代理限制的原始钱包。
- **B1（仅金额）：** 仅执行每笔交易金额检查（maxAmountPerTx）。模拟简单的支出上限。
- **B2（双重检查）：** 金额检查 + 时间窗口检查。模拟带有营业时间的支出上限。
- **B3（完整门控，无升级）：** 应用全部八项策略检查，但已拒绝的操作会被静默丢弃（无人工审核，无升级）。模拟没有人在回路的严格自动化过滤器。
- **AESP（完整）：** 全部八项策略检查 + 三级升级。已拒绝的操作会被路由到 ReviewManager；自动批准、明确审核和生物识别层级均已启用。

3）指标：对于 H1（安全），我们将自动拦截性能报告为主要安全结果，并将升级视为运营负载：

- **自动拦截率：** 在无人类参与的情况下，被策略门确定性地已拒绝的攻击请求所占比例。这是有保障的防御：自动拦截的请求无论人类行为如何都无法继续。
- **升级负载率：** 通过 ReviewManager 路由至人类主体的攻击请求所占比例。这是

[3]该yault_前缀反映了参考结算层集成（Yault）。面向其他结算层的部署可以重命名这些工具，同时保留相同的JSON Schema接口。[4] https://github.com/ya-xyz/aesp

一项运营负担指标，不计入确定性安全成功。

- **假阳性率（FPR）：**被自动拦截或升级处理的合法请求所占比例。

- **逐项检查归因：**在消融研究中，用于确定八项检查中哪一项触发了自动拦截。对于 H2（效率），我们测量：

- **策略评估延迟：**执行 8 项检查序列（evaluatePolicy()）所需的时间。
- **签名延迟：**Ed25519 和 secp256k1 的签名时间（通过密码学模块完成）。
- **密钥派生延迟：**HKDF 上下文隔离派生所需的时间。

- **端到端开销：**从操作请求到签名授权的总延迟（策略检查 + 签名 + 派生）。

全部延迟测量均使用 performance.now()，丢弃 100 次预热迭代，并进行 $\geq$1,000 次测量迭代。结果将报告为 $\geq$5 次独立试验的中位数及四分位距（IQR）。主要基准环境为现代 Node.js 运行时（例如 v25），并通过 wasm-bindgen 加载 WASM 绑定；在适用情况下，也将报告使用原生C FFI绑定的对比测量结果。

4）消融设计：为单独评估每项策略检查的作用，我们每次从完整的八重检查门（B3 配置，不含升级）中移除一项检查，并重新评估自动拦截率。预期输出为一张增量表，展示每项检查对整体安全覆盖率的边际贡献。

B. 计划分析

1）安全覆盖率（*H1*）：基线比较将报告五种配置（B0–B3 和 AESP Full）各自的自动拦截率、升级负载率和 FPR。我们预计，八重检查门将自动拦截绝大多数单条件违规，而升级处理负载将集中在那些通过单项检查、但呈现可疑跨请求模式的聚合违规上。消融分析将量化每项检查的边际贡献。

解读升级。升级并不意味着自动拦截。如果人工审核者未经检查就批准所有升级的请求，实际保护效果就仅等于自动拦截率。因此，我们仅将升级作为审核负荷报告，并单独分析其可用性影响（第 XIII-D节）。

2) 延迟开销（*H2*）：我们将测量以下各项的单次操作延迟：8 项检查的策略评估、预算跟踪器查询、Ed25519 签名、secp256k1 签名、EIP-712 类型化数据签名、HKDF 上下文派生、SHA-256 哈希（密码学模块与 Web Crypto 回退），以及复合端到端路径。所有密码学操作均以 WASM 绑定作为主要目标进行基准测试；原生 C 外部函数接口的测量结果将作为次要对比数据报告。假设阈值为端到端开销中位数 200 毫秒；鉴于单项密码学操作通常只需几毫秒至十几毫秒即可完成，我们预计实际开销会显著低于此值。

C. 交易不可关联性

为了评估隐私子系统，我们将生成 $N = 1,000$ 笔交易，涉及多个代理，每个代理都使用上下文隔离的地址（隐私级别 ℓ_2），并对由此产生的链上足迹应用模拟的地址聚类启发式方法（共同输入所有权和时间接近性）。

实验将比较三种配置：（1）不采取归集对策（基线）；（2）仅采用时间抖动（±30%）；（3）抖动 + Fisher-Yates 洗牌 + 批量归集（批次大小为 5，批次间延迟为 10–60 分钟）。我们预计，这些配置的交易可关联性将逐步降低，而残余可链接性主要源于归集输出中的基于金额的关联。

D. 预期的有效性威胁

构念效度。攻击语料库为合成数据；现实世界的攻击分布可能有所不同。我们通过对全部八种检查类型进行分层，并纳入同时违反多个条件的复合攻击，来缓解这一问题。

外部效度。延迟测量结果将因平台而异。WASM 的性能在不同运行时（浏览器、Deno、服务器端 Node.js）中有所不同，原生 FFI 的性能也会与 WASM 的结果不同。由于即时编译预热，基于浏览器的部署可能会产生 1.5–2$\times$ 倍的额外开销，而原生 C 外部函数接口部署预计会更快。

审批疲劳。本评估假设人类主体会对审核请求作出最优响应。实际上，大量审核请求可能会降低人类决策质量。通过速率限制和智能批处理来应对审批疲劳，是未来工作（参见第 XV节）。

XIV. 案例研究

为展示 AESP 的表现力和实用性，我们提出三个案例研究，分别运用了协议功能的不同组合。每个案例研究都通过 AESP SDK 实现为端到端场景。

A. 三方杂货配送

此场景模拟消费者代理、杂货店代理和配送代理协同完成一笔杂货配送交易。消费者代理通过 negotiationFSM（第 VII节）与杂货店代理协商价格，形成 EIP-712 承诺（第 VIII节），并协调配送付款。涉及的关键协议功能：

- 多代理身份派生（来自不同主体的三个代理）

- 通过报价／还价／接受流程进行跨代理协商
- 附带托管的双方签署承诺
- 策略执行（支出限额、链限制）
- 首笔高额杂货订单的人工审核

B. 自主云资源优化器

该场景模拟一个自主管理云基础设施支出的AI智能体。该代理会监控资源利用率并协商竞价实例价格，但须遵守人类定义的预算策略。运用的关键协议特性：

- 带预算跟踪功能的策略引擎（日/周/月限额）
- 关键策略变更分类（提高预算需进行生物识别）
- 紧急冻结/解冻周期
- 滚动预算窗口管理

C. 隐私保护型*NFT*猎手

此场景模拟一个代理在多个交易市场中搜寻并购买NFT，同时保护买家的身份隐私。演示的协议关键功能包括：

- 全部三种隐私级别（透明、基础、隔离）
- 地址池补充与领取
- 采用 Fisher-Yates 洗牌和时间抖动的批量归集
- 通过密码学模块进行上下文隔离地址派生
- 加密审计标签归档

这些案例研究表明，AESP 的模块化设计使不同应用能够按需组合协议功能：杂货场景侧重多方协作，云优化器侧重策略执行和人工监督，而 NFT 搜索器则侧重隐私。

第十五章：讨论与局限

评估状态。 第 十三章 详细说明了用于验证这两项假设的完整评估方法——请求语料、基线、指标和消融设计。该框架将确定性保护（自动拦截率）与运维负担（升级处理负载）区分开来，并测量策略评估、密码学签名和密钥派生的端到端延迟开销。

结算层依赖。 AESP 定义协议并生成签名授权，但不实现链上执行。端到端运行需要一个符合规范的结算层。当前实现旨在与 Yault 集成，但在架构上不依赖特定结算层。

隐私局限与残余泄漏。上下文隔离隐私依赖于支持 REV32 模式下基于 HKDF 的上下文派生的钱包。如第 XIII-C节所述，归集对策（时间抖动、Fisher-Yates 洗牌、批量归集）旨在降低地址关联性，但预计归集输出中仍会存在由基于金额的相关性导致的残余泄漏。具体而言，如果对手能够观察到整合交易金额，并将其与已知的临时地址余额相匹配，则部分重新关联便成为可能。在归集过程中采用金额拆分或混币等技术可以减少这一残余泄漏，但目前尚未实现。

审批疲劳。requireReviewFirstPay 检查与预算边界接近度预计会产生不可忽略的假阳性率。在持续的高吞吐操作下，这可能引发审核疲劳。AESP 目前支持紧急级别和到期期限，但尚未实现速率限制、对审核请求的智能批处理，或根据人工审批模式进行学习的自适应阈值。解决这一问题是未来工作的优先级。

绑定性能差异。 密码学延迟会因绑定目标而异。由于 JIT 编译预热和更严格的内存限制，浏览器中的 WASM 执行开销可能比 Node.js 高出 1.5–2× 。使用原生 C 外部函数接口或 Dart FFI 绑定的服务器端和移动端部署无需经过 WASM 层，预计可达到原生 Rust 性能。

后量子迁移。 虽然密码学模块中已支持 ML-DSA-44（涵盖所有绑定目标），但要完成后量子迁移，还需要采用后量子密钥协商（替代 X25519）和后量子加密（以基于后量子 KEM 的对称方案 + 替代 AES-256-GCM）。目前的后量子支持仅涵盖签名。

可扩展性。 审核队列会导致超出策略范围的操作出现延迟。在高吞吐量的代理经济中，默认 30 分钟的审核期限可能过长。自适应截止时间策略和批量审核界面是有待改进的方面。

形式化验证。 虽然协商有限状态机和策略评估序列都有明确的规范，但其实现尚未经过形式化验证。下一步自然是对状态机和策略评估逻辑应用模型检查。

监管合规。 AESP 的策略引擎和审计轨迹可满足合规要求（支出限额、允许列表、审计记录），但针对特定司法管辖区的法规（例如，汇款机构分类、MiCA）仍需在结算层和平台层进行分析。

XVI. 结论

我们提出了 AESP，这是一种使人工智能代理能够参与经济交易，同时确保经济主权始终归人类主体所有的协议。该协议通过确定性策略评估、分级的人机协同审核、双重签名的 EIP-712 承诺、基于 HKDF 的上下文隔离隐私，以及构建于 ACE-GF 底层架构之上并提供多平台绑定的全栈密码学基础，确保代理具备经济能力，但绝不拥有经济主权这一不变量。

我们制定了一套完整的评估方法，包括四种限制程度逐步提高的基线、逐项检查的消融设计，以及一项交易不可链接性实验，旨在检验两个可证伪假设：（H1）八重检查门自动拦截 ≥90% 的未授权交易，假

阳性率为 {v2}%，同时单独报告升级处理负载；（H2）端到端延迟开销低于 200 毫秒。

AESP以近期研究确立的概念基础为基础，尤其是 Google DeepMind 的智能 AI 委派框架 [28]，并对其中提出的要求给出了具体且已实现的方案。配套的开源 TypeScript SDK（包含 208 项测试和 10 个模块）[5]支持与 MCP 和 A2A 互操作。随着人工智能代理越来越有能力自主开展经济行为，能够以量化方式确保人类主权的协议将成为关键基础设施。AESP 为这一新兴需求提供了经过形式化规范且可实现的基础。

参考文献

[1] D. Minarsch、S. A. Hosseini、M. Favorito 与 J. Ward，“自主经济智能体作为区块链的二层技术：框架介绍与用例演示”，载于第*2*届加密谷区块链技术会议（*CVCBT*）论文集，2020年，第27–35页。[2] S. Alqithami，“区块链上的自主智能体：标准、执行模型与信任边界”，*arXiv:2601.04583*，2026年1月。[3] M. Xu，“智能体经济：面向自主人工智能代理的区块链基础”，*arXiv:2602.14219*，2026年2月。[4] B. A. Hu、Y. Liu 与 H. Rong，“无信任自治：理解自主主权去中心化AI智能体中的动机、收益与治理困境”，*arXiv:2505.09757*，2025年5月。[5] P. F. Christiano、J. Leike、T. B. Brown、M. Martic、S. Legg 与 D. Amodei，“基于人类偏好的深度强化学习”，载于*NeurIPS*，第30卷，2017年。[6] L. Ouyang等，“利用人类反馈训练语言模型遵循指令”，载于*NeurIPS*，第35卷，2022年。[7] E. C. Cheng等，“迈向安全且负责任的人工智能代理：面向透明度、问责与可信性的三支柱模型”，*arXiv:2601.06223*，2026年1月。[8] B. A. Hu 与 H. Rong，“主权智能体：迈向去中心化AI中的基础设施主权与分散问责”，*arXiv:2602.14951*，2026年2月。[9] R. S. Sandhu、E. J. Coyne、H. L. Feinstein 与 C. E. Youman，“基于角色的访问控制模型”，*IEEE Computer*，第29卷，第2期，第38–47页，1996年。[10] V. C. Hu等，“基于属性的访问控制（ABAC）定义与考量指南”，NIST SP 800-162，2014年。[11] J. B. Dennis 与 E. C. Van Horn，“多道程序计算的程序语义”，*ACM*通讯，第9卷，155，1966年 3，143期 页 –。[12] N. Abaev、D. Klimov、G. Levinov、D. Mimran、Y. Elovici 与 A. Shabtai，“AgentGuardian：学习访问控制策略以治理AI智能体行为”，*arXiv:2601.10440*，2026年1月。[13] A. G. Ganie，“保障人工智能代理安全：面向工业应用实施基于角色的访问控制”，*arXiv:2509.11431*，2025年9月。[14] T. Wahrstatter、M. Solomon、B. DiFrancesco 与 V. Buterin，“ERC-5564：隐匿地址”，以太坊改进提案，2022年。[15] M. Nadler 与 F. Schar，“Tornado Cash与区块链隐私：面向经济学家与政策制定者的入门读本”，圣路易斯联邦储备银行评论，第105卷，第2期，第122–136页，2023年。[16] G. Maxwell，“CoinJoin：面向现实世界的比特币隐私”，BitcoinTalk论坛，2013年8月。[17] F.-X. Wicht、Z. Wang、D. V. Le 与 C. Cachin，“区块链隐私的交易级模型”，载于金融密码学与数据安全（*FC*）会议论文集，2024年。[18] R. Bloemen、L. Logvinov 与 J. Evans，“EIP-712：类型化结构化数据哈希与签名”，以太坊改进提案，2017年。[19] H. Krawczyk 与 P. Eronen，“基于HMAC的提取-扩展密钥派生函数（HKDF）”，IETF RFC 5869，2010年。[20] A. Asgaonkar 与 B. Krishnamachari，“解决买卖双方困境：一种无需可信中介、可证明防作弊的数字商品交付与支付双押金托管智能合约”，载于*IEEE ICBC*会议论文集，2019年。

[21] Anthropic，“模型上下文协议（MCP）规范”，版本2025年11月25日，2025年。[22] Google，“代理间协议（A2A）”，版本0.3，2025年。[23] M. Sporny、A. Guy、M. Sabadello 与 D. Reed（编），“去中心化标识符（DIDs）v1.0”，W3C 推荐标准，2022年。[24] M. Sporny、D. Longley 与 D. Chadwick（编），“可验证凭证数据模型 v1.1”，W3C 推荐标准，2022年。[25] C. Allen，“迈向自我主权身份之路”，2016年4月。[26] E. Krul、H.-Y. Paik、S. Ruj 与 S. S. Kanhere，“SoK：信任自主主权身份”，载于隐私增强技术研讨会（*PETS*）论文集，2024年。[27] S. Srivastava 与 J. B. Bullock，“人工智能、全球治理与数字主权”，*arXiv:2410.17481*，2024年10月。[28] N. Tomašev等，“智能 AI 委托”，Google DeepMind，*arXiv:2602.11865*，2026年2月。[29] J. S. Wang，“ACE-GF：面向自主数字实体的原子化无种子存储身份原语”，*arXiv:2511.20505*，2025年11月。[30] 美国国家标准与技术研究院，“基于模格的数字签名标准（ML-DSA）”，FIPS 204，2024年8月。[31] T. M. Fernández-Caramés 与 P. Fraga-Lamas，“迈向后量子区块链：抗量子计算攻击的区块链密码学综述”，*IEEE Access*，第8卷，第21091–21116页，2020年。

附录A 协议常量

表IX 汇总了 AESP 全文使用的协议常量。

表IX：协议常量

常量	值	章节
REV32上下文命名空间	ACEGF-REV32-V1-* + 上下文标签	IV
最大层级深度	5	IV
策略范围等级	1, 2, 3, 10	VI
最大协商轮数	10	VII
协商生存时间	24小时	VII
审核截止时间	30 分钟	IX
地址池大小	5	X
合并间隔	4 小时	X
合并抖动	±30%	X
合并批次	5	X
批次间延迟	10–60 分钟	X
审计批次阈值	50	X
审计时间窗口	5 分钟	X
EIP-712 域名	“YalletAgentCommitment”	VIII
EIP-712 版本	“1”	VIII
Argon2id 内存	4 MB	V
Argon2id 迭代次数	3	V
HKDF 信息前缀	ACEGF-REV32-V1-	V
支持的链	7（参见 §V-C）	V

附录 B
密码学模块 API 摘要

Y

表 X 列出了 AESP TypeScript 软件开发工具包使用的主要导出项。所示函数名称来自 WASM 绑定（`wasm.rs`）；`_wasm` 后缀是绑定层的命名约定，并不表示其行为特定于 WASM——对应的函数也通过 C 外部函数接口（`ffi.rs`）和 Dart FFI 绑定提供，并遵循各自的平台命名约定（例如，C 外部函数接口使用`_ffi` 后缀）。

[5]https://github.com/ya-xyz/aesp

表 X：主要密码学模块导出项（WASM绑定名称）

导出	函数
view_钱包 _统一 _with_上下文 _wasm	在以下上下文中派生多链钱包上下文
以太坊虚拟机_签名_类型化_数据_与_上下文	带上下文隔lation
Solana_签名_交易_与_上下文	Solana 交易签名
acegf_加密_用于_xidentity	为接收方身份加密数据
acegf_计算_dh_密钥_wasm	X25519 ECDH 密钥协商
acegf_密封_口令_wasm	Argon2id 口令封装
acegf_解封_口令_wasm	口令解封
pqc_签名_消息_wasm	ML-DSA-44 后量子签名
pqc_验证_签名_wasm	ML-DSA-44 验证