人工智能代理注册解决方案概述

Aditi Singh, Abul Ehtesham, Ramesh Raskar, Mahesh Lambe,
Pradyumna Chari, Jared James Grogan, Abhishek Singh, Saket Kumar

Project NANDA

arxiv:2508.03095v1 中译本

ABSTRACT

随着自主 AI 代理在云端、企业级和去中心化环境中扩展，需要标准化的注册系统来支持发现、身份识别和能力共享的需求变得至关重要。本文综述了三种由独特元数据模型定义的突出注册方法：MCP 的 mcp.json、A2A 的 Agent Card 以及 NANDA 的 AgentFacts。MCP 使用 GitHub 认证发布的集中式元注册表，并采用结构化元数据进行服务器发现。A2A 通过基于 JSON 的 Agent Cards 实现了去中心化的交互，这些卡片可以通过知名 URI、精选目录或直接配置来发现。NANDA Index 引入了 AgentFacts，这是一种旨在实现动态发现、凭据能力以及跨域互操作性的加密可验证且保护隐私的元数据模型。本文从四个维度对比了这三种方法：安全性、扩展性、认证和维护性。文章最后提出了建议和推荐措施，以指导未来 AI 代理互联网注册系统的设计与采用。

1 介绍

自主 AI 代理正在迅速成为从云原生助手和机器人到去中心化系统和基于边缘的物联网控制器等各个领域的重要基础。这些代理独立行动，做出决策，并在大规模上进行协作。随着代理群体在异构平台和管理边界中增长至数十亿，识别、发现并实时信任代理的能力已经成为一个关键的基础设施挑战。

传统的机制如 DNS 和静态服务目录并不适合代理生态系统，这些系统需要动态发现、可验证元数据和隐私保护交互。遗留系统假设固定的端点和基于所有权的信任模型，缺乏灵活性和加密保证，而这些是用于旋转能力、改变位置和形成短暂协作的代理所必需的。

为了应对这些限制，一些代理框架引入了发现元数据模型。本文重点讨论三种新兴方法：

本文对四种注册架构进行了比较分析，每种架构都提供了应对这些挑战的独特方法：

- **MCP 注册表** (Anthropic, 2025) [1]，该系统提供了一个使用结构化 `mcp.json` 文件的集中式元数据层，用于代理发现和安装。
- **A2A 代理卡片** [2] 描述了代理能力及端点的标准 JSON 格式，使通过众所周知的 URL、精选目录或私有分发进行发现成为可能。
- **Microsoft Entra 代理 ID** (微软, 2025)[3]：一种通过 SaaS 交付的企业级目录，内置在 Azure AD 中，提供生命周期管理、治理和零信任控制功能。
- **NANDA 指数：代理事实** , (MIT AIDE, 2025) [4] 一种去中心化的、可加密验证的格式，支持动态解析、凭证断言和隐私保护查询。

与其调查一般代理通信协议 [5]，这项工作是对这些人工智能注册解决方案的集中比较。它探讨了每种方法如何支持实时发现、身份验证和跨域互操作性。

论文组织如下：第二节概述了代理注册表的背景和动机。第三节介绍了评估框架和功能标准。第四至第六节深入探讨了 MCP、A2A、Microsoft Entra Agent ID 和 NANDA。第七节提供了安全、可扩展性、认证和维护性的比较分析。第八节总结了设计建议，并为多代理系统中的注册表采用提供了实际建议。

通过在新兴代理基础设施需求背景下比较这些注册表，本调查突出了当前的差距和推动人工智能代理互联网未来发展的新兴解决方案。

2 背景

现代网络运行在一个反应式的、由客户端驱动的模型中，在这种模型中，服务会等待外部请求后再作出响应。尽管云自动化和事件驱动设计取得了显著进展，但这种架构仍然在很大程度上不足以应对新兴的人工智能代理互联网这一范式转变，在这个转变中，自主的目标导向软件代理代表用户进行协商、协调，并主动采取行动。与通常无状态且生命周期较短的传统网络资源不同，自主的人工智能代理是能够发起控制流、保留长期记忆、动态适应环境并生成下属代理的持久计算实体。这些代理需要支持高频更新、实时身份解析以及跨异构系统和组织边界进行可信元数据交换的基础设施。

这种转变给发现和协调带来了重大挑战。当前基于 DNS、IP 地址和证书机构构建的互联网堆栈并未设计为处理数万亿个快速移动的自主代理。撤销延迟、状态传播、身份验证和路由规模的限制都成为关键瓶颈。

为了填补这些空白，新的注册模型正在出现，它们从静态的名称解析系统转向为自主代理量身定制的动态、元数据丰富的发现层。本文关注三种这样的模型，每个模型都配有一个独特的元数据架构：

- **MCP 注册表：** 一个集中式元注册表，通过 `mcp.json` 支持结构化代理元数据发布，为与 MCP 兼容的代理提供安装和版本支持。
- **A2A 代理卡片：** 一种灵活的、分散式的代理自我描述格式，通过众所周知的 URL、精选注册表或配置文件实现发现。
- **NANDA 代理事实：** 一种旨在动态解析、凭据能力声明和联合环境的加密可验证且保护隐私的元数据模式。

这些注册系统旨在解决 AI 代理互联网的基本需求：亚秒级身份解析、模式验证的能力表示、可验证的信任模型以及隐私感知的发现。每个模型提供了不同的架构立场，集中式、联邦式或去中心化，来满足这些要求。本调查将这三种方法置于更广泛的网络基础设施转型中进行定位，与从拨号到宽带和从 IPv4 到 IPv6 等转变的历史平行。通过理解现有系统的局限性和 AI 代理的独特需求，我们强调为什么专门为 AI 设计的注册系统对于可扩展、安全且互操作的代理生态系统至关重要。

2.1 设计评估维度

为了将候选注册表架构与上述要求进行比较，我们从四个核心维度进行评估。

- **安全：** 通过加密签名确保注册表记录和元数据的完整性。抵御欺骗、注册表中毒和中间人攻击。
- **认证：** 发布者身份验证机制（例如，GitHub OAuth + DNS-TXT，DID-VC 签发，X.509 PKI）。注册表更新的控制方式以及如何强制执行命名空间所有权。
- **可扩展性：** 通过基于 TTL 的缓存、联合部署或 CDN 卸载来处理高查询量和大量代理群体的能力。支持低延迟、地理分布式的解析。
- **维护：** 操作简便性：模式优先设计，核心代码最少，解耦的元数据托管。升级、迁移路径的简易性以及通过避免托管可执行代码来减少补丁表面。

这些维度提供了一个结构化、基于来源的评分标准，用于第 4 – 9 节中的比较分析。

3 MCP 注册表

MCP 注册表是一个集中式的 “元注册表”，用于发现和安装 MCP 服务器。发布者通过一个执行 GitHub OAuth 流程的 CLI 工具推送版本化的 `mcp.json`，对于反向 DNS 命名空间，则进行 DNS TXT 挑战。基于 Go 的 REST API 暴露了读取端点（无需身份验证）和写入端点（GitHub OAuth + DNS 验证），将原始 JSON 存储在对象存储中，在 MongoDB 中索引元数据（带有内存回退），并生成异步作业或 webhooks。下游 MCP 客户端应用程序轮询注册表（或私有镜像），本地缓存数据，并为最终用户提供服务，而无需直接调用中心服务。

3.1 安全

注册表仅接受经过身份验证的 GitHub 身份提供的元数据，对于域范围的命名空间，则来自经过 DNS 验证的域名。它不托管可执行代码；相反，它只保存元数据，继承已建立注册表（npm、PyPI、DockerHub）的代码级安全性。这最大程度地减少了攻击面，并将身份验证和域控制委托给已证明可靠的系统。

3.2 认证

所有发布请求都需要一个与提交用户或组织关联的 GitHub OAuth 承载令牌。对于反向 DNS 命名空间（例如 `com.microsoft`），需要提供 DNS TXT 记录证明并链接到该 GitHub 身份。读取操作公开可访问。

3.3 可扩展性

只有少量的 MCP 客户端应用程序查询中央注册表；它们缓存并为数百万最终用户提供数据。该 API 支持异步处理和 webhooks。元数据存储在适合灵活文档式记录的 MongoDB 中，并通过可 CDN 缓存的 HTTP 端点提供服务；可选中间层（私有镜像、精选源）可以分担负载。

3.4 维护

注册表的核心服务由 `mcp.json`（OpenAPI/JSON Schema）驱动，无需维护包托管或扫描。一个 CLI 工具自动化了发布和验证流程。模式更新独立于服务代码进行，并且验证逻辑由发布者负责。

4 代理 2 代理（A2A）协议

Agent2Agent (A2A) 协议是一种传输无关的企业级标准，用于异构系统之间的代理间通信。A2A 允许潜在不透明、特定供应商或闭源的自主代理使用安全 HTTP 传输上的共享 JSON-RPC 接口进行发现、协商和协作。

A2A 优化了异步、长运行时间、多模式和流式交互，支持代理之间灵活的任务移交，而无需了解内部执行模型。通过其声明式的 AgentCard，它实现了技能、能力和认证要求的动态发现，建立了一个代理间协作的标准模型。

4.1 安全

A2A 依赖传输层安全性 (TLS) 和已建立的网络安全性最佳实践。身份和认证通过标准 HTTP 标头在 A2A JSON-RPC 负载之外处理，允许与 OAuth2、API 密钥和 mTLS 兼容。服务器身份通过 TLS 证书进行验证，而客户端根据 AgentCard 中宣传的安全方案进行身份验证。推送通知（webhooks）使用设置期间协商的每个客户端凭证、令牌或方案进行身份验证。代理不共享内部状态；交互范围限于声明的能力，并通过任务和工件管理，减少攻击面并限制数据暴露。

4.2 认证

AgentCards 明确使用 OpenAPI 风格的安全方案声明支持的身份验证机制（例如，Bearer 令牌、OpenID Connect、API 密钥）。客户端必须通过带外方式获取凭据，并将它们包含在请求头中。每个 RPC 调用都会单独进行身份验证，当凭据缺失或无效时，服务器会返回带有指导信息的 HTTP 401/403 响应。在执行过程中，如果需要辅助凭据（例如，代理工具访问），任务将转换为需要认证状态，并且客户端将在后续消息中提供所需的凭据。

4.3 可扩展性

A2A 的基于任务的、无状态传输通过 HTTP 和 SSE 实现了分布式代理系统中的水平扩展。代理在 AgentCards 中以声明方式定义其能力，允许注册表和发现服务动态编目可用的服务。任务是具有唯一 ID、状态更新和工件流的长期存在对象。通过 SSE 进行的流式传输和推送通知减少了轮询开销。任务生命周期支持细粒度事件（已提交、正在处理、需要输入等），即使在易出错环境中也能实现响应性和鲁棒性的编排。

4.4 可维护性

A2A 是有意设计为简单且可扩展的，基于 HTTP 和 JSON-RPC 2.0 构建。它尽量减少自定义逻辑并避免特定协议。功能如 AgentCard 和结构化数据格式（例如，TextPart、DataPart、FilePart）确保消息的一致解释同时允许多样性模态。模式演化是灵活的：代理可以根据技能定义能力，覆盖输入/输出 MIME 类型，并动态扩展其 AgentCard。任务处理和消息结构遵循一致且可扩展的惯例。

5 Microsoft Entra 代理 ID

Microsoft Entra Agent ID 提供了一个管理型的企业级目录，用于 AI 代理身份。在 Copilot Studio 或 Azure AI Foundry 中创建的代理将自动出现在 Entra 管理中心中的“Agent ID”应用程序中。身份管理人员可以使用与用户或服务身份相同的工具和策略来查看、管理和治理这些非人类身份的身份生命周期。即将推出的功能包括最低权限令牌发放、扩展的条件访问以及跨租户身份联合。一旦技术文档和运营数据可用，将能够进一步分析 Microsoft Entra Agent ID 的安全性、认证能力、可伸缩性和可维护性。

图 1: Microsoft Entra 代理 ID 概览 [3]

6 NANDA 指数

网络代理和分散式人工智能（NANDA）索引，设想为一种跨越平台、组织和协议的代理、资源和工具注册表的拼接结构，提供了一种精益且模块化的架构，用于在去中心化环境中发现代理。这种类似拼接的索引支持纳入 NANDA 原生代理和第三方代理，通过统一框架实现广泛的可发现性。为规模、隐私和互操作性设计，NANDA 将静态标识符解析与动态代理元数据分开，以支持联邦代理生态系统中的快速发现、凭据验证和灵活路由。

通过这种方法，NANDA 指数实现了代理的全球互操作性、可发现性和适应性治理，而无需实施集中控制。NANDA 指数并不是作为普遍权威存在，而是允许商业、政府和个人利益相关者选择其代理如何与索引交互。代理可以直接列在 NANDA 指数中，或通过重定向到外部平台进行引用。这确保了实体在其自己的领域内保有对访问政策、信任管理和认证流程的完全控制权，同时仍然参与全球互联的人工智能生态系统。

设计的核心概念是最小的 AgentAddr 记录，这是一个 $Ed25519$ 签名对象，将代理标识符映射到一个或多个可验证的元数据位置：一个公共 FactsURL，一个可选的隐私保护 PrivateFactsURL，以及一个用于实时路由的 AdaptiveRouterURL。这些记录轻量级（≤ 120 字节），可以缓存且稳定，在高流动环境中的注册表写入也最少。

完整的发现流程结构分为三个模块化层次，每个层次都针对可扩展的去中心化代理生态系统中的特定角色进行了优化：

1. **精益索引层**：提供了一个去中心化的、可缓存的映射，用于将代理标识符映射到签名记录（代理地址记录，≤ 120 字节），其中包括元数据 URL、加密签名和 TTL。这些轻量级且防篡改的记录作为发现和路由的不可变锚点，无需频繁写入索引。
2. **代理事实层**：使用自描述的 JSON-LD 文档（作为 W3C 可验证凭据签名）分发丰富的、模式验证过的元数据。这些文档描述了功能、端点、遥测和认证逻辑，可以托管在代理控制的域或去中心化存储（例如 IPFS）中，允许独立于索引进行隐私保护更新。
3. **动态分辨率层**：解释元数据以通过静态、旋转或自适应端点路由查询。这一层支持地理位置感知的路由、负载均衡以及短暂的端点轮换（TTL 5 – 15 分钟），以满足大规模性能、隐私和弹性要求。

6.1 安全

NANDA 通过端到端的加密保证来确保安全。每个代理地址都使用 $Ed25519$ 由原始注册表进行签名，确保真实性、完整性和不可变性。AgentFacts 文档被签署为 W3C 可验证凭据 (VC v2)，支持短期凭证（通常<5 分钟）并通过 VC-状态列表实现实时撤销。这个加密框架防止了身份欺骗、能力伪造或代理交互过程中的冒充行为。此外，使用私有事实网址支持保护隐私的解析路径，有助于模糊客户访问模式，符合零信任安全原则。

6.2 身份验证

代理元数据通过去中心化标识符（DIDs）进行认证，AgentFacts 中所有声明，如能力、合规性断言或审计结果，必须由锚定在可验证信任域的凭证权威机构签名。元数据更新不需要对精益索引层进行修改；相反，它们是由代理或授权的第三方基础设施独立托管和签名的。这支持具有可验证来源的去中心化发布。

精益索引层不中介实时认证。相反，它充当密码学信任根，为下游验证提供不可变的引用。代理发布者可以使用基于 DID 的凭据、发行者声明或支持自我主权和企业对齐代理生命周期的委托授权模型进行身份验证。

6.3 可扩展性

NANDA 通过其分层和模块化设计实现了互联网规模的性能。精益索引层仅存储签名指针（代理地址），从而最小化写入开销，而元数据和操作状态则由代理事实层和动态解析层外部管理。这种分离消除了写入放大并支持水平扩展。

基于 TTL 的缓存确保了解析请求由边缘解析器或客户端高效处理，减少了对轻量级索引层的压力。同时，使用自适应解析器 URL 和旋转端点池实现了响应式的实时路由，而不会给索引基础设施带来负担。

系统支持联合部署，其中精益索引实例可以是特定领域的（如医疗保健、金融）或地理位置相关的。跨实例互操作性通过基于 DID 的解析和可验证声明实现，允许在联合信任边界内无缝运行。

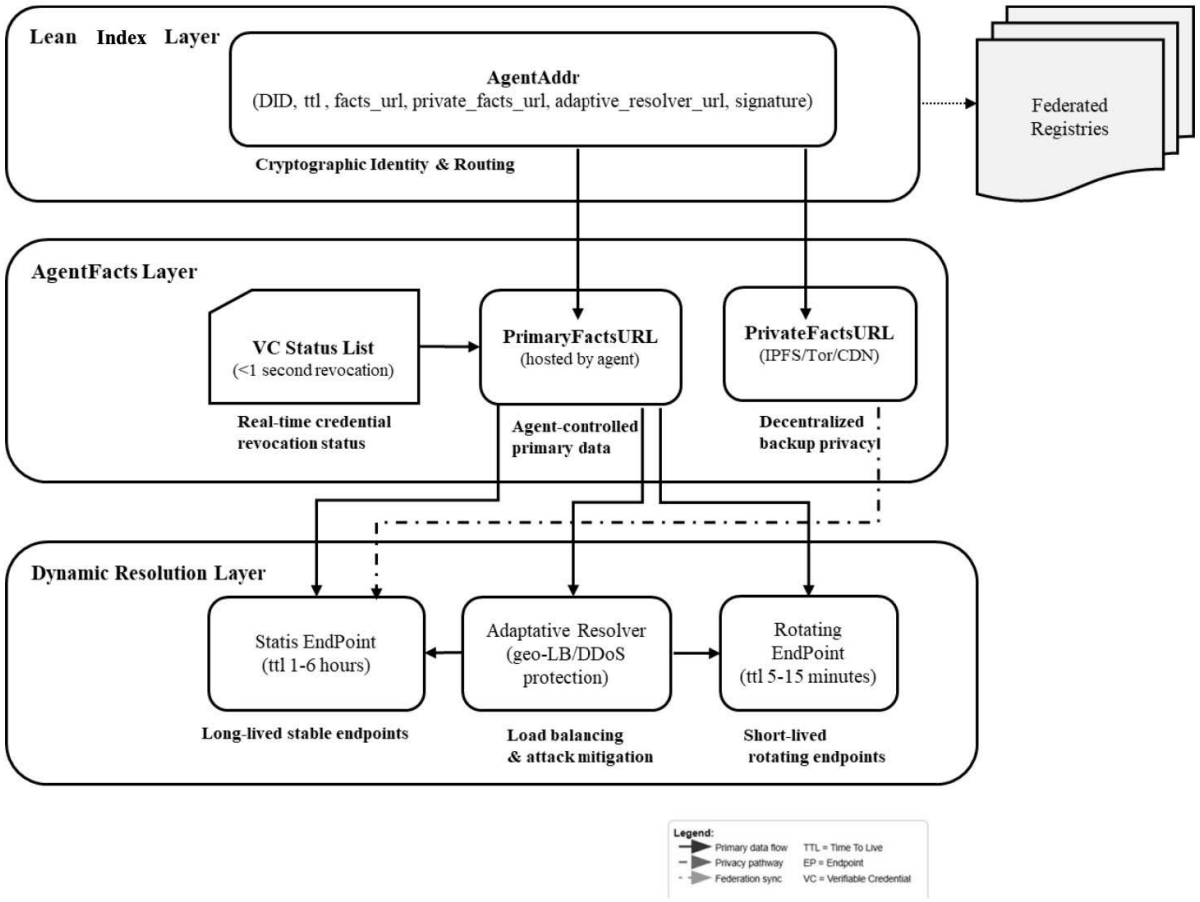

图 2: NANDA 指数和 AgentFacts 架构：一个用于去中心化 AI 代理发现和路由的模块化三层系统。**精益索引层**解析代理标识符为包含加密身份、元数据 URL 和路由信息的已签名代理地址记录，这些记录在注册表之间进行联合。**代理事实层**通过主要和私有 URL 分发动态且由代理控制的元数据，并实时更新可验证凭据（VC）状态以支持撤销。**动态分辨率层**通过稳定、适应性和轮换策略实现端点解析，以支持隐私保护、负载均衡和 DDoS 抵御能力。这种分层设计使跨联合域的代理发现具有可扩展性、尊重隐私并能互操作 [6]。

6.4 可维护性

精益索引层的逻辑被设计得尽量简单和稳定，从而减少操作复杂性和出错的可能性。由于动态状态和能力是通过外部 AgentFacts 管理的，因此索引更新不频繁且轻量级。

所有记录均符合基于版本化的 JSON-LD 的模式，并带有向前兼容的上下文，确保未来可扩展性。元数据演变不需要更改 Lean Index Layer，从而在代理级别实现快速迭代。用于验证、路由策略和凭证检查的支持工具是模块化且外部引用的，确保长期多方生态系统中的可维护性。

7 比较评价

我们的比较分析是两方面的：

- 表 1 比较了来自 MCP、A2A（谷歌）和 NANDA Index 的注册表解决方案，分析了它们的目的、发现路径、信任原语、隐私机制、端点新鲜度策略、模式复杂性和最佳适用场景。

表 1: A2A、MCP 和 NANDA 指数的 AI 代理注册解决方案对比

维度	麦克白	谷歌 A2A	NANDA 指数
目的	Enable publication and discovery of agent servers for LLM tool use	Advertise a server-side agent's endpoint and skills for JSON-RPC	Convey live endpoints plus cryptographically-signed capabilities, SBOM hash, privacy path, revocation info
发现路径	REST: `/v0/服务器`, `/v0/servers/{id}`	One-step fetch at `/.well-known/agent.json` or central catalogue	两步：轻量级注册表 → 事实网址 /私有事实网址
信任原语	Bearer token + GitHub credentials	Plain HTTPS and optional token, self-declared attributes	W3C Verifiable Credential v2 signatures; VC-Status revocation (<1 s), support for third-party audited attributes
隐私选项	Container metadata and manifest publishing; optional offline operation	None (lookup hits agent host)	可选的私有事实网址通过 IPFS/Tor；隐藏请求者
端点新鲜度	Timestamps on publish: 创建时间, 更新于; dynamic via REST	Assumes minutes-to-hours stability (no TTL field)	TTL-scoped lists: static (1–6 h) or rotating (5–15 m)
模式权重	Approx. 1–3 KB JSON	≈ 0.3–1 KB JSON	1–3 KB JSON-LD + VC
最佳匹配用例	Centralized registry of agent/server tools (Docker/NPM); designed for scalable deployment and LLM plug-in integration	Stable SaaS agents inside a single marketplace	Highly mobile, privacy-sensitive, or safety-critical agents at trillion scale

- 表 2 提供了代理卡片（用于 A2A）和代理事实（用于 NANDA 索引）之间的详细特征级别比较，突出了关键差异——包括但不限于元数据结构、端点建模、加密保证和可扩展性。

8 代理注册架构的分阶段演化

代理注册系统已经从简单的基于文件的描述发展到具有结构化发现协议的分布式、可加密验证的注册表。本节概述了跨越三个关键阶段的发展历程，每个阶段都增加了互操作性、可扩展性、信任和治理的层次。理解这些阶段有助于阐明现有系统的设计理念以及在每个成熟度级别上所做的权衡。

8.1 第一阶段——静态、孤立发现

最早的注册机制依赖于在代理域中知名位置发布的静态文件（例如，JSON 或 YAML 清单）。这些文件主要由人工或紧密耦合的运行时消费，并包含最少的静态元数据。常见属性包括代理名称、端点和基本功能。

示例： 谷歌 A2A 的`/.well-known/agent.json`，

8.2 第二阶段——动态 RESTful API

此阶段通过 HTTP API 引入了运行时内省，并正式验证了 JSON 模式。MCP 具有 RESTful API，可以在客户端应用程序中查找和列出可用的 MCP 服务器。

表 2: 详细特征比较：代理卡与代理事实

特征	代理卡（Google A2A）	代理事实（NANDA 指数）
标识符	url (host-tied)	Id
标签与描述	name, description, version	agent name, label, description, version
提供者信息	provider.organization, url	provider.{name,url,id}
静态端点	single url	"端点. 静态"[ ]
旋转/地理端点	–	"端点. 旋转"[ ] (URL + TTL)
适配器路由器	–	"endpoints.adaptive_router".url
技能 / 能力	技能 [], 能力。	技能 [] + 能力
认证方案	安全方案, 安全	认证能力
协议标志	能力. 流式传输, 推送通知	能力. 流式传输
软件物料清单/ 完整性	–	sbomHash (in VC)
遥测	–	遥测.{端点, 采样}
性能/审计	–	评估块
隐私路径	–	私有事实网址
加密包装器	plain JSON, HTTPS	JSON-LD + W3C VC
	optional	signature
ttl/新鲜度	none (HTTP cache)	per-endpoint TTL fields
大小	0.3 – 1 KB	1 – 3 KB
获取跳数	1 (well-known)	2 (registry → Facts)
撤销速度	HTTP re-fetch (min)	VC-Status (<1 s)
实现工作量	low	higher (JSON-LD, VC Toolchain)

8.3 第三阶段——可验证元数据和联邦信任及 AI 代理拼图

此阶段的注册表采用接近纳达索引的加密验证和联邦信任机制。代理元数据使用 W3C 可验证凭据（VCs）、PKI 证书或 JSON 规范化与哈希签名进行签署。基于 ID 的身份或域绑定签名确保真实性，而 TTL 和撤销机制实现细粒度缓存控制。这些设计实现了信任传递性、审计性和代理间验证。它们非常适合移动、隐私敏感或安全关键部署。这种方法接近纳达索引。

9 结论

自主 AI 代理在企业、研究和消费领域中的泛滥产生了紧迫的基础设施挑战：如何在互联网规模上发现、识别并信任这些代理。本调查研究了四种代表性的注册架构：MCP 元注册表、A2A、微软 Entra Agent ID 和 NANDA 索引，每种架构通过独特的设计理念来满足安全、认证、可扩展性和维护的核心需求。

我们的分析揭示了将塑造代理发现基础设施未来发展的几个关键见解：

1. **架构折衷取决于协议**。“正确”的注册表架构在很大程度上依赖于部署环境。具有现有 Azure AD 基础设施的企业环境得益于 Entra Agent ID 的无缝集成和零维护方法。开放研究社区和去中心化应用需要 NANDA Index 提供的加密保证和联盟治理。像 MCP 这样的协议特定生态系统受益于利用现有身份验证系统（GitHub OAuth）的同时保持简单性的专门构建注册表。

2. **去中心化实现长期可持续性**。虽然集中式方法提供了操作上的简便性，但它们创造了单一故障点和供应商锁定风险，这些风险随着代理生态系统的发展而变得越来越成问题。NANDA 索引的联邦设计展示了去中心化架构如何实现可扩展性和社区治理。NANDA 中的双路径解析模式特别解决了隐私问题，这些问题将在代理交互增多时变得更加关键。
3. **安全必须内置，而不是附加**。所有审查过的注册表都认识到，无论通过 W3C 可验证凭据（NANDA）、DNS-TXT 验证（MCP）还是 Azure AD 的企业安全控制（Entra），密码完整性都是基础。然而，由 NANDA 开创的将静态身份解析与动态元数据分发分离提供了防止注册表中毒和实现隐私保护发现模式的最稳健基础。
4. **互操作性仍然是关键差距**。尽管它们的架构存在差异，这些注册表服务于重叠的用例，并且随着代理生态系统的成熟，不可避免地需要互操作。跨协议发现、统一命名空间管理以及便携式代理身份代表了注册表基础设施开发的下一个前沿。
5. **社区治理对生态系统健康至关重要**。历史表明，最具有弹性的互联网基础设施如 DNS、HTTP 和电子邮件，都是从开放的多方利益相关者治理中产生的。虽然像 Entra Agent ID 这样的专有平台服务于特定的企业需求，但更广泛的代理生态系统需要由社区管理的注册表，这些注册表可以独立于任何单一供应商的利益而发展。

参考文献

[1] Model Context Protocol Contributors. Model Context Protocol Registry. `https://github.com/modelcontextprotocol/registry`, 2025. Accessed: 2025-07-22.

[2] Google Developer Blog. Announcing the agent2agent protocol (A2A). `https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/`, 2025. Accessed: 2025-07-22.

[3] Alex Simons. Announcing microsoft entra agent id: Secure and manage your ai agents. `https://techcommunity.microsoft.com/blog/microsoft-entra-blog/announcing-microsoft-entra-agent-id-secure-and-manage-your-ai-agents/3827392`, May 2025. Accessed: 2025-05-29.

[4] Project NANDA. Upgrade or switch: The need for new registry architecture for the internet of ai agents. `https://github.com/aidecentralized/nandapapers`, 2025. Accessed: 2025-07-22.

[5] Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). `https://arxiv.org/abs/2505.02279`, 2025.

[6] Ramesh Raskar, Pradyumna Chari, John Zinky, Mahesh Lambe, Jared James Grogan, Sichao Wang, Rajesh Ranjan, Rekha Singhal, Shailja Gupta, Robert Lincourt, Raghu Bala, Aditi Joshi, Abhishek Singh, Ayush Chopra, Dimitris Stripelis, Bhuwan B, Sumit Kumar, and Maria Gorskikh. Beyond dns: Unlocking the internet of ai agents via the nanda index and verified agentfacts, 2025.