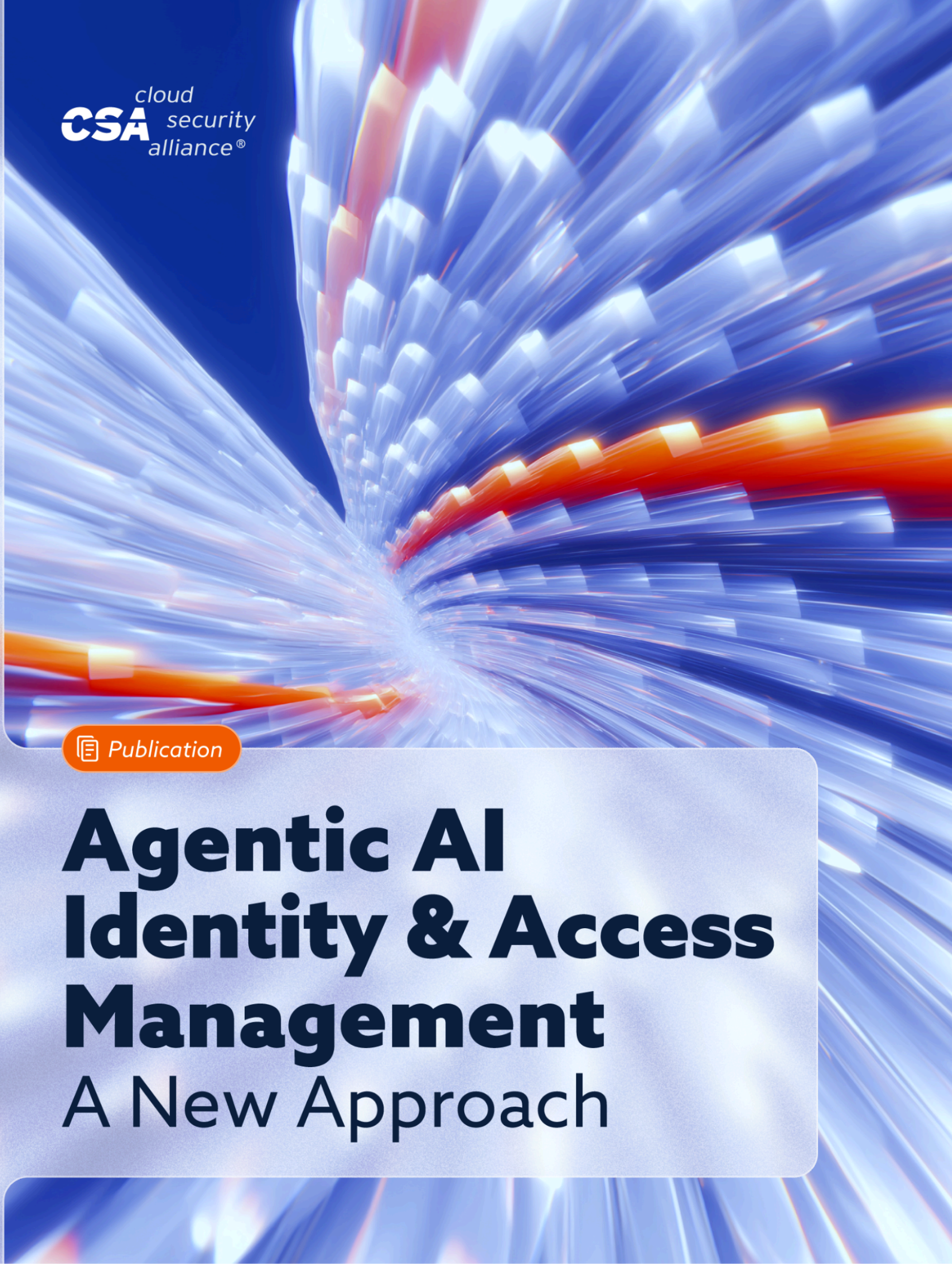
cloud
CSA security
alliance®
Publication
Agentic AI
Identity & Access
Management
A New Approach

The permanent and official location for the AI Safety Initiative is https://cloudsecurityalliance.org/ai-safety-initiative

# Acknowledgments

## Co-Authors

Ken Huang
Vineeth Sai Narajala
John Yeoh
Jason Ross
Mahesh Lambe
Ramesh Raskar
Youssef Harkati
Jerry Huang
Idan Habler
Chris Hughes
Akram Sheriff
Staford Titus

## CSA Global Staff

John Yeoh

### Graphic Design

Stephen Lumpe
Stephen Smith

## Contributors/Reviewers

John Jiang
Elad Luz
Syed Aamir
Eray ALTILI
Bhavin Jethra
Yuanji Sun
Suhas M
Govindaraj Palanisamy (Govi)
Victor Ronin
Pratyush Mishra
Ramesha Reddy
Ashwin Sharma
Michael Morgenstern
Vatsal Gupta
Josephine
Anirudh Murali
Mahesh Kukreja
Estevenson Solano
Harpreet Singh
Schandrasekhar Varma
Sheetal Kawle
Mohsin Khan
Krishna R Maddikara
Rajiv Dewan
Nirupam Samanta
Jayesh Dalmet
Josephine Liu

# Premier AI Safety Ambassadors

CSA proudly acknowledges the initial cohort of Premier AI Safety Ambassadors. They sit at the forefront of the future of AI safety best practices, and play a leading role in promoting AI safety within their organization, advocating for responsible AI practices and promoting pragmatic solutions to manage AI risks.


airia
Enterprise AI Simplified

Airia is an enterprise AI full-stack platform to quickly and securely modernize all workflows, deploy industry-leading AI models, provide instant time to value and create impactful ROI. Airia provides complete AI lifecycle integration, protects corporate data and simplifies AI adoption across the enterprise.

Deloitte.

The Deloitte network, a global leader in professional services, operates in 150 countries with over 460,000 people. United by a culture of integrity, client focus, commitment to colleagues, and appreciation of differences, Deloitte supports companies in developing innovative, sustainable solutions. In Italy, Deloitte has over 14,000 professionals across 24 offices, offering cross-disciplinary expertise and high-quality services to tackle complex business challenges.

ENDOR LABS

Endor Labs is a consolidated AppSec platform for teams that are frustrated with the status quo of “alert noise” without any real solutions. Upstarts and Fortune 500 alike use Endor Labs to make smart risk decisions. We eliminate findings that waste time (but track for transparency!), and enable AppSec and developers to fix vulnerabilities quickly, intelligently, and inexpensively. Get SCA with 92% less noise, fix code 6.2x faster, and comply with standards like FedRAMP, PCI, SLSA, and NIST SSDF.

Microsoft prioritizes security above all else. We empower organizations to navigate the growing threat landscape with confidence. Our AI-first platform brings together unmatched, large-scale threat intelligence and industry-leading, responsible generative AI interwoven into every aspect of our offering. Together, they power the most comprehensive, integrated, end-to-end protection in the industry. Built on a foundation of trust, security, and privacy, these solutions work with business applications that organizations use every day.

Reco leads in Dynamic SaaS Security, closing the SaaS Security Gap caused by app, AI, configuration, identity, and data sprawl. Reco secures the full SaaS lifecycle—tracking all apps, connections, users, and data. It ensures posture, compliance, and access controls remain tight as new apps and AI tools emerge. With fast integration and real-time threat alerts, Reco adapts to rapid SaaS change, keeping your environment secure and compliant.

# Table of Contents

# Abstract

Agentic AI Identity and Access Management (IAM) represents a fundamental paradigm shift from traditional identity management systems. Unlike conventional IAM protocols designed for predictable human users and static applications, agentic AI systems operate autonomously, make dynamic decisions, and require fine-grained access controls that adapt in real-time. Traditional protocols like OAuth 2.1 and SAML fall short due to their coarse-grained, static nature, inability to handle machine-speed authentication, and lack of contextual awareness required for autonomous AI operations.

The solution requires a comprehensive framework combining zero-trust architecture, decentralized identity management, dynamic policy-based access control, and continuous monitoring to ensure secure, accountable, and compliant AI agent deployments.

Traditional Identity and Access Management (IAM) systems, primarily designed for human users or static machine identities via protocols such as OAuth, OpenID Connect (OIDC), and SAML. However, these systems are fundamentally inadequate for the dynamic, interdependent, and often ephemeral nature of AI agents operating at scale within Multi Agent Systems (MAS) – a computational system composed of multiple interacting intelligent agents that work collectively.

This paper posits the imperative for a novel Agentic AI - IAM framework:

1. We deconstruct the limitations of existing protocols when applied to MAS, illustrating with concrete examples why their coarse-grained controls, single-entity focus, and lack of context-awareness render them ineffective.

2. We then propose a comprehensive framework built upon rich, verifiable Agent Identities (IDs), leveraging Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs), that encapsulate an agent's capabilities, provenance, behavioral scope, and security posture.

Our framework includes an Agent Naming Service (ANS) for secure and capability-aware discovery, dynamic fine-grained access control mechanisms (ABAC, PBAC, JIT), and critically, a unified global session management and policy enforcement layer for real-time control and consistent revocation across heterogeneous agent communication protocols. We also explore how Zero-Knowledge Proofs (ZKPs) enable privacy-preserving attribute disclosure and verifiable policy compliance.

We outline the architecture, operational lifecycle, innovative contributions, and security considerations of this new IAM paradigm, aiming to establish the foundational trust, accountability, and security necessary for the burgeoning field of agentic AI and the complex ecosystems they will inhibit.

**Keywords:** Agentic AI, Identity Management, Access Control, Multi-Agent Systems, Decentralized Identifiers, Verifiable Credentials, Zero-Knowledge Proofs, AI Security, Zero Trust.

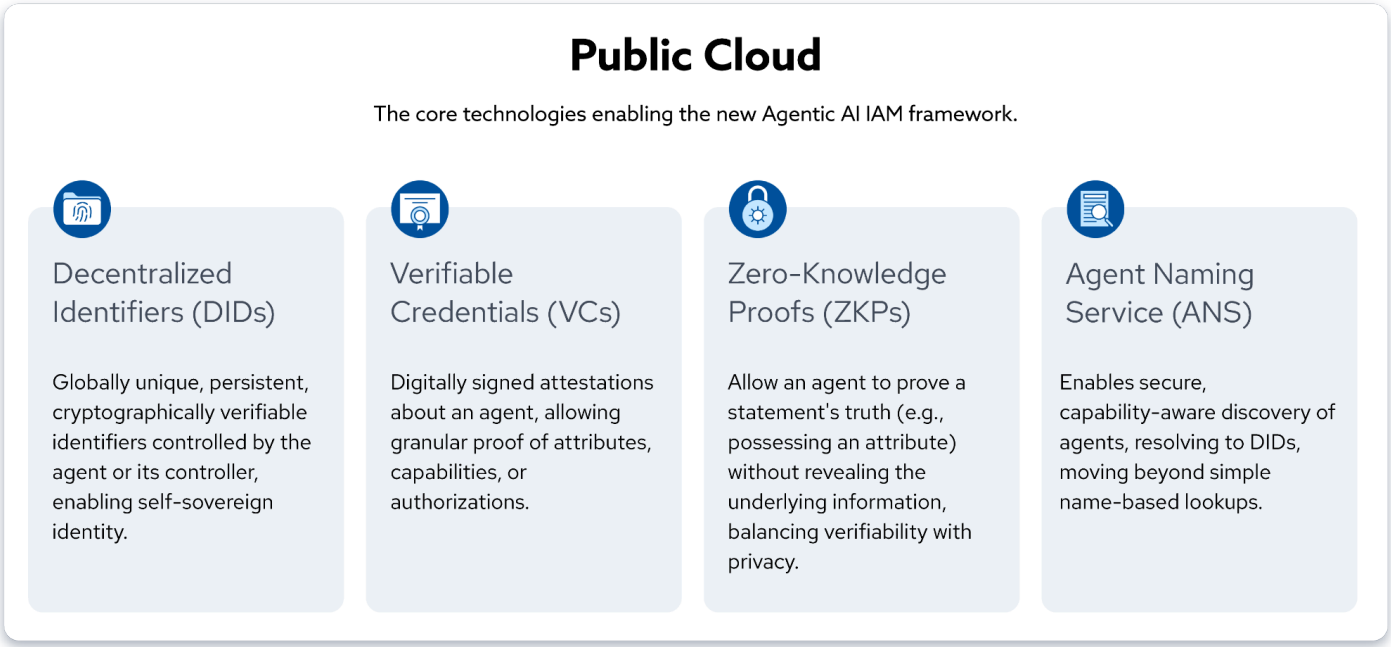


*Figure 1: Public Cloud*

# 1. Introduction

## Understanding Agentic AI and Its Unique Identity Challenges

### What Makes Agentic AI Different

Agentic AI systems represent a new class of autonomous software entities that can plan, reason, and execute complex multi-step tasks with minimal human supervision. Unlike traditional applications that follow predetermined workflows, AI agents:

- Operate with unprecedented autonomy, making real-time decisions based on contextual information
- Interact across multiple systems simultaneously, often requiring different permission levels for each interaction
- Adapt their behavior dynamically based on learned patterns and environmental changes
- Scale to thousands of agents within enterprise environments, each requiring individual identity management

## The Scale and Complexity Challenge

Organizations are rapidly adopting agentic AI, with 60% of enterprises expected to involve AI agents within a year. (Recent survey by Allganize in US) This explosive growth creates several critical challenges:

**Identity Explosion:** Organizations face managing tens of thousands of agent identities instead of hundreds of human users, with each agent requiring distinct authentication and authorization profiles.

**Dynamic Permission Requirements:** AI agents need permissions that change moment-by-moment based on context, risk levels, and mission objectives, far exceeding the capabilities of static role-based systems.

**Mixed Identity Scenarios:** Agents may operate both as autonomous entities with their own credentials and as delegates acting on behalf of human users, creating complex authorization chains.

Do we need a new approach for Agentic AI Identity Management? The failure to address the unique identity challenges posed by AI agents operating in Multi-Agent Systems (MAS) could lead to catastrophic security breaches, loss of accountability, and erosion of trust in these powerful technologies. For instance, without robust agent-specific IAM, a compromised autonomous agent in a financial system could cascade unauthorized transactions, or a swarm of interacting agents in critical infrastructure could be manipulated with devastating consequences. This paper builds on our earlier Cloud Security Alliance publication (Huang, 2025a), expanding the scope and proposing a more comprehensive framework tailored to the needs of agentic AI.

The core problem this current paper addresses is the fundamental mismatch between existing IAM paradigms (e.g., OAuth 2.1, OpenID Connect OIDC, SAML) and the unique characteristics of AI agents in MAS. These agents exhibit autonomy, ephemerality, dynamically evolving capabilities, complex trust relationships, and may soon be operating at an unprecedented scale. Their actions carry direct consequences, demanding robust accountability. If not managed appropriately, delegated authority can cascade through multiple agents, obscuring responsibility. The European Union's AI Act (European Parliament and Council, 2023) and similar regulatory initiatives underscore the growing societal demand for transparency, accountability, and human oversight in AI systems, making robust agent IAM an unavoidable prerequisite.

Real-world indicators of these limitations are already emerging. As reported in the Wall Street Journal, Rosenbush (2025) notes that AI agents consistently encounter challenges when interacting with APIs and services designed around human-centric authentication flows and session models. These operational constraints reinforce the need for identity architectures tailored to autonomous, non-human actors.

Inspired by preliminary discussions on IDs for AI systems (Chan et al., 2024b), this paper also examines the limitations of current IAM protocols in MAS settings and illustrates through concrete examples how their coarse-grained permissions, single-entity assumptions, limited inclusion of Non-Human Identities (NHIs) and lack of contextual adaptability fall short. We then expound the need for a new, holistic Agentic AI IAM framework. We contend that merely adapting existing protocols is insufficient. Instead, a purpose-built approach is required, one that redefines agent identity, incorporates novel cryptographic primitives, and establishes new mechanisms for discovery, layered authentication, access control, and real-time policy enforcement tailored to the agentic paradigm.

This paper makes the following contributions:

- It critically analyzes the inadequacies of traditional IAM protocols (OAuth, OIDC, SAML) in the context of MAS, providing concrete examples of their failure points.
- It defines the essential components of a rich, verifiable, and dynamic AI Agent Identity (ID), leveraging Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs).
- It proposes a layered Agentic AI IAM architectural framework incorporating DIDs, VCs, Zero-Knowledge Proofs (ZKPs), an Agent Naming and Discovery Service (ANS), dynamic access control models, and a novel unified global session management and policy enforcement layer.
- It details how this framework addresses the lifecycle of agent IAM, from identity creation and attestation to runtime authorization, logging, monitoring, and incident response.
- It compares centralized, decentralized, and federated deployment models for this framework, offering guidance on their applicability, and analyzes security considerations using the MAESTRO framework.

The remainder of this paper is structured as follows: Section 2 elaborates on the imperative for a new agentic IAM paradigm by dissecting the limitations of traditional IAM. Section 3 defines the multifaceted nature of an AI agent's identity. Section 4 presents the proposed Agentic AI IAM framework architecture. Section 5 discusses the operational use cases of Agent IDs within this framework. Section 6 analyzes deployment models and governance. Section 7 details security considerations. Section 8 highlights the innovative contributions. Section 9 discusses future work, and Section 10 offers conclusions and summation.

# 2. The Imperative for a New Agentic IAM Paradigm

## The Evolution of AI Agents: From Simple Tools to Digital Teammates

AI agents are rapidly advancing, moving from single-task tools to sophisticated, integrated digital colleagues. This evolution can be understood in three key stages:

### 1. Interactive Agents

The foundational stage consists of agents designed for narrow, repetitive tasks. They operate by calling on a specific tool to execute a well-defined command.

- **Focus**: Single-task execution.
- **Example:** A chatbot that answers asked questions from a knowledge base and searches the internet or internal knowledge stores.

## 2. Autonomous Agents

This next phase introduces complexity and strategic planning. Autonomous agents work within multi-agent systems to achieve broader, goal-oriented objectives. They collaborate with other agents, create plans, and use their own distinct toolsets to reach a shared goal.

- **Focus:** Complex, multi-step problem-solving.
- **Example:** A system where one agent researches travel options, another finds the best price, and a third books the itinerary.

## 3. Digital Employees

The current frontier is the Digital Employee—an AI that emulates human decision-making and integrates deeply into organizational structures. More than just a tool, it is a learning-driven collaborator.

- **Focus:** Dynamic, human-like collaboration and continuous learning.
- **Key Traits:**
    - Integrated: Part of the HR system with a defined role and team access.
    - Collaborative: Works seamlessly with both humans and other agents.
    - Adaptive: Learns from interactions and new context to improve performance.
    - Resourceful: Can not only use existing tools but also create new ones to solve novel problems.

## Summary of an Agent's Journey

| Stage | Scope | Collaboration | Capability |
|---|---|---|---|
| Interactive Agent | Single, defined task | None (human-triggered) | Uses a specific tool |
| Autonomous Agent | Complex goal | Agent-to-agent | Plans and uses a set of tools |
| Digital Employee | Team member role | Human and agent | Learns, adapts, and creates new tools |

*Table 1: Evolution of AI Agent Capabilities*

To power the future of Agentic AI, we must upgrade our identity standards and systems to govern how agents securely access data and act across all our systems—from APIs to sensitive business processes.

The emergence of MAS necessitates a fundamental rethinking of how we manage identity and access management paradigm. While traditional IAM protocols have been sufficient for human-centric and

simpler machine-to-machine interactions via service accounts or workload identities, their core assumptions and mechanisms break down when faced with the complexities of autonomous, interacting AI agents.

## 2.1 Revisiting Traditional IAM

The authorization model of OAuth 2.1, designed for user-delegated tasks, falls short for the emerging class of autonomous agents. As agents become more capable, they introduce new requirements that current standards don't address:

- Dynamic Permissions: Permissions must be highly granular, easily revokable, and fully auditable.
- Cross-Boundary Communication: Agents need to interact securely with other agents, even across different organizational trust boundaries.
- Fluid Ownership: The system must handle scenarios where agent ownership changes dynamically.

Driving changes in existing identity standards and systems to support these capabilities is essential for building the secure, compliant foundation that enterprises need to adopt in the new autonomous agentic AI.

### Critical Shortcomings of Traditional IAM Protocols

#### OAuth 2.1 Limitations in Agentic Environments

OAuth 2.1, while effective for human-centric applications, faces significant limitations when applied to AI agents:

**Coarse-Grained Scopes**: OAuth's static scope model cannot accommodate the fine-grained, resource-specific permissions that AI agents require. For example, an agent might need access to "all photos from last week" or "SELECT * FROM my_emails WHERE sender LIKE '%@microsoft.com'" - permissions that cannot be expressed in traditional OAuth scopes.

**Lack of Agent Identity Recognition:** Current OAuth implementations do not distinguish between human users and AI agents, making it impossible to apply agent-specific policies or track agent actions separately.

**Inadequate Delegation Support:** OAuth 2.1 lacks mechanisms for secure, traceable delegation where agents can act on behalf of users while maintaining clear accountability chains.

**Static Trust Model:** OAuth assumes that once authenticated, an entity remains trustworthy throughout the session(Trust is time-scoped via the token lifetime - usually 1h, not usage-scoped) - an assumption that fails with AI agents that may be compromised or manipulated through adversarial attacks.

The following is the table to summarize the security limitation of oAuth in Aentic AI workflow.

| Functionality | Security Limitation of OAuth | Consequences in Agentic workflows |
|---|---|---|
| Agent Identity | No sub-agent identity model | Hard to track or authorize individual agents |
| Token Security | Tokens are bearer-based, transferable | Risk of impersonation, leakage |
| Contextual Authorization | Scopes are static | Poor fit for adaptive or dynamic AI agent behavior |
| Delegation and Trust | No native delegation between agents | Breaks multi-agent workflows |
| Identity Federation | Requires OIDC, no strong trust signals | Identity spoofing or confusion in authorization |

*Table 2: OAuth 2.1 Security Limitations in Agentic AI Workflows*

## SAML's Fundamental Incompatibility

SAML's limitations are even more pronounced in agentic environments:

**XML-Based Overhead:** SAML's heavy reliance on XML-based assertions creates performance bottlenecks for machine-speed authentication requirements, where AI agents may need to authenticate 148 times more frequently than human users. (Okta's benchmarks show AI workloads initiate 148x more authentication requests per hour than humans.)

**Session-Based Authentication:** SAML's session-oriented approach conflicts with AI agents' need for continuous, real-time authentication and authorization decisions.

Static Attribute Model: SAML's predefined user attributes cannot capture the dynamic, contextual factors that should influence AI agent access decisions.

## The Confused Deputy Problem

Traditional IAM systems create significant confused deputy vulnerabilities where AI agents may gain access to resources that should not be available to them but are accessible to the system they're running on. This occurs because:

- Agents inherit broad system permissions rather than user-specific, constrained permissions
- There's no clear distinction between what the agent can access versus what the user has authorized
- Audit trails fail to capture the true scope of agent actions and their authorization basis

## SSO Integrations

Enterprises expect to manage all their software, including AI agents using MCP, through a centralized Single Sign-On (SSO) system. This allows administrators to control user access and licensing from a single identity provider (IdP).

The current method for connecting an AI agent like Claude to other enterprise apps (e.g., Google Drive, Slack) is flawed for two main reasons:

- **Poor User Experience:** To connect each external app, an employee must go through a repetitive series of redirects to authenticate and provide consent via OAuth prompts. This process is tedious, especially as the number of integrated applications grows.
- **Lack of Enterprise Control:** The connections between applications are established directly, bypassing the central IdP. This means enterprise administrators have no visibility or control over these data-access permissions. See related MCP RFC at Github: https://github.com/modelcontextprotocol/modelcontextprotocol/pull/284
  In a corporate environment, the decision to allow an AI agent to access company data should be made by IT administrators, not individual employees. This practice creates "unchecked interactions between third-party services and firms' sensitive internal resources," a concern highlighted by security officers

## Emerging Threats and Attack Vectors

### Tool Poisoning and Manipulation Attacks

The integration of AI agents with external tools through protocols like MCP introduces new attack vectors:

- **MCP Preference Manipulation:** Attackers can deploy customized MCP servers that manipulate AI agents to prioritize malicious tools over legitimate ones, potentially leading to economic exploitation or data theft.
- **Tool Squatting:** Malicious actors can create tools with names similar to legitimate services, tricking AI agents into using compromised alternatives.
- **Rug Pull Attacks:** Legitimate-appearing tools can suddenly change behavior or disappear, leaving agents and users vulnerable to data loss or service disruption.

### Prompt Injection and Semantic Attacks

AI agents face unique vulnerabilities through prompt injection attacks that can manipulate agent behavior:

- Cross-prompt injection enables attackers to include malicious instructions in documents or emails that agents process

- Memory poisoning allows bad actors to corrupt agent memory with false information that influences future decisions
- Cascading hallucinations can cause agents to generate and reinforce incorrect outputs over time

## The Urgent Need to Update OAuth 2 for the Age of Autonomous Agents

OAuth 2 has served us well for today's task-focused agents that operate on behalf of users. However, as AI agents evolve to become more autonomous and capable, we're encountering a critical gap in current authorization frameworks. These advanced agents demand a new set of requirements: **more granular, dynamic, and easily revocable permissions**, along with robust audit trails. They also need to securely interact with other agents across various trust boundaries and seamlessly handle changes in ownership. To unlock the full potential of these agents for enterprises, we must evolve existing standards to ensure compliance and maintain data security.

### Key Changes Required for OAuth 2

We've identified several crucial areas where OAuth 2 needs to adapt:

- **Recognize Agent IDs as First-Class Actors:** Agents require their own distinct identity within the OAuth model, separate from clients. When an agent registers with an Identity Provider (IdP) or accesses a resource, it should be able to clearly identify itself as an agent. Furthermore, we need a standardized way to represent interactions when a computer-using agent accesses a resource through a client.

- **Standardize Permission Models for Agents:** Agents should possess their own defined set of privileges, rather than simply proxying a user's rights. This allows for more precise control over their actions.

- **Ensure Transparent and Traceable Agent Actions:** It's essential to clearly distinguish when an agent is acting:

    - On behalf of a user.
    - On its own behalf.
    - On behalf of another agent or a chain of agents. This clarity is paramount for forensic analysis, policy enforcement, and building trust in agent operations.

- **Enable Permission Discovery and Delegation:** Agents should have the ability to discover the permissions necessary to complete a task and then request them. This request could come directly from the user, an upstream agent, or through a chain of upstream agents ultimately linked back to the user.

- **Support Fine-Grained, Resource-Specific, Least-Privilege Access:** The current OAuth scopes model needs updating to support more precise control over resource access. This includes:

- **Collections of resources:** Such as "all photos from last week."
- **Nodes in a hierarchy:** For example, "all files in the `/taxinfo` directory."
- **Specific classes or categories:** Like "high business impact" or "confidential" data.
- **Query-based access:** Enabling permissions for something like "SELECT * FROM my_emails WHERE sender LIKE '%@microsoft.com'."
- **Individual resources:** Such as `{customer_ID, 12345}`.

These targeted updates will provide users and organizations with the essential controls, visibility, specificity, and granularity needed to confidently embrace the transformative potential of AI agents.

Protocols such as OAuth 2.1 (Hardt, 2012), OpenID Connect (OIDC) (OpenID Foundation, 2014), and SAML (OASIS, 2005) are ubiquitous for authentication and authorization across diverse environments. Alongside these, foundational enterprise protocols such as Kerberos for domain authentication and LDAP for directory services (as well as comprehensive cloud identity solutions) form the backbone of current identity management for human users and traditional IT systems. While useful for those contexts, their design is misaligned with the requirements of MAS.

### 2.1.1. Use of Conventional IAM in Human-Scoped Agent Workflows

In limited contexts, particularly involving single agents or direct human-to-agent platform interactions, these traditional protocols and systems can still play a role, primarily in managing the human interface to agentic systems or bootstrapping initial agent context. This includes scenarios where a human user initiates an AI agent to act as their personal assistant, executing tasks on their behalf. Even in such delegated scenarios, the AI agent should still be instantiated with and utilize verifiable identities (DIDs and VCs) to ensure accountability, auditability, and secure access to resources and services:

- **Human Authentication to Platforms:**
    - **OIDC and SAML for Web/Federated Access:** A human user authenticating to an AI agent deployment platform via OIDC or SAML is a standard use case. This is often federated through broader cloud identity solutions like **Microsoft Entra ID**, which can manage both cloud-native and synchronized enterprise identities. For instance, a developer logging into an AI orchestration platform would use their enterprise OIDC or SAML provider.
    - **Kerberos for Enterprise Internal Access:** Within many corporate networks, Kerberos remains the primary mechanism for authenticating human users to internal services and platforms. A developer or operator might authenticate to their workstation and subsequently to an agent management console using Kerberos.
    - The Kerberos protocol uses secret-key cryptography to ensure that credentials are never sent over the network in plaintext. Instead, it relies on tickets to authenticate users and services, which helps to protect against eavesdropping and replay attacks.
- **Deriving Initial Agent Context and Attributes:**
    - LDAP as an Attribute Source: Enterprise LDAP directories (such as those underpinning Active Directory, often managed or federated by Microsoft Entra ID in hybrid

environments) serve as authoritative sources for user attributes and group memberships. This information can be used by an organization to issue initial Verifiable Credentials (VCs) to an agent, attesting to its ownership, departmental affiliation, or preliminary set of permissions derived from the human deployer's context.
    - As an example, the platform may then spawn agents that initially operate under a context derived from this human user's authenticated session. The platform then creates an agent, for example, mcp-dev-agent, which is used by developers and might initially inherit some basic permissions tied to the developer's identity (sourced via OIDC, SAML, or Kerberos, with attributes potentially enriched from LDAP) to access specific code repositories and documentation systems.

- **OAuth 2.1 for Simple Delegated Access by a Single Agent:** An AI agent acting as a client can use OAuth 2.1 to access a resource server on behalf of a human user who has granted explicit consent. This mirrors traditional third-party application access. If mcp-dev-agent needs to retrieve additional project context using Model Context Protocol (MCP) to better understand the developer's codebase, it would go through a standard OAuth 2.1 flow, obtaining an access token scoped specifically to read project documentation and code structures that the developer has authorized.
- **NHI Tasks and Automations:** NHIs may inherit access permissions from the human who deployed them. Service account and automation scripts are often granted access through IAM role inheritance, static credential issuance or predefined group membership. These identities, while non-autonomous and task-specific, are typically predictable, constrained, and managed through traditional IAM protocols

    - **Service Accounts and OAuth 2.1**: Traditional NHIs like service accounts often rely on OAuth 2.1 client credentials flows to authenticate to cloud APIs or internal services. These flows are compatible with existing identity governance platforms, though they lack behavioral awareness and session integrity.
    - **Secrets and Certificates as Surrogate Authentication**: Static secrets and certificates issued through PKI or secret management systems are effective in authentication but lack real-time behavior verification without add on protocols, traceability, and support in dynamic environments.
    - **Role-Based Access Tied to Humans**: NHIs in many organizations are indirectly managed by assigning them roles or permissions derived from human owners or creators (e.g., LDAP group inheritance or IAM role mapping). This makes sense for simple automation tools but fails in autonomous systems. AI agents may retain excessive privileges long after their human creators change roles or leave the organization, creating persistent security gaps. More importantly, autonomous systems require dynamic, context-aware permissions that adapt to changing operational needs—something static human-derived roles cannot provide. When agents need to collaborate, escalate privileges, or operate across different security domains, the rigid inheritance model breaks down entirely, leaving organizations exposed to both over-privileged access and operational failures.

However, these scenarios typically involve a single, well-defined agent acting in a relatively static role, often directly tethered to a human user's session or a pre-configured machine identity derived from these traditional IAM systems. The complexities, and the breakdown of these approaches, arise when multiple agents interact autonomously, as detailed next.

### 2.1.2. Fundamental Insufficiencies for Multi-Agent Systems (MAS)

The dynamic, decentralized, and deeply interconnected nature of MAS exposes critical flaws in traditional IAM:

- **Coarse-Grained and Static Permissions:** OAuth and SAML primarily rely on pre-defined scopes or roles that are often too broad and static for the fluid operational needs of AI agents. Agents in MAS frequently require granular, task-specific permissions that can change dynamically based on context, mission objectives, or real-time data analysis.
    - *Example:* Consider a disaster response MAS
        - Agent-Search (locates survivors via drone_feed_api) might initially need read-only access to map data (map.read) and drone telemetry (drone.telemetry.read).
        - Upon finding a survivor, it might need to delegate a task to Agent-MedicalDispatch (coordinates medical_resources_api), which then requires access to medical_assets.request and hospital_availability.query.
        - Agent-Search might then also need to alert Agent-Logistics (manages supply_chain_api) about resource needs, requiring supply.request permissions.
    - In this example, traditional OAuth scopes (read_all_data, manage_all_resources) would lead to massive over-privileging, while re-authenticating for every micro-permission change is untenable.
- **Single-Entity Focus vs. Complex Delegations:** These protocols are architected around a single authenticated principal (user or application). They struggle to model and secure complex delegation chains where an agent might spawn sub-agents, or where an agent acts on behalf of multiple principals simultaneously (e.g., a user and an organization).
    - *Example:* A user (userAlice_DID) delegates a financial planning task to Agent-Planner (agentPlanner_DID).
      Agent-Planner determines it needs specialized market analysis and spawns Agent-MarketAnalyst (agentMarketAnalyst_DID) and tax optimization from Agent-TaxOptimizer (agentTaxOptimizer_DID).
        - How is userAlice_DID's authority securely and granularly passed from Agent-Planner to its sub-agents?
        - Does Agent-MarketAnalyst inherit all of Agent-Planner's (and thus userAlice_DID's) permissions, or just the bare minimum for market data access?
    - OAuth's delegation (e.g., token exchange) is typically designed for simpler scenarios and doesn't provide a clear, auditable chain of fine-grained delegated authority. As a result, using the OAuth model, accountability becomes blurred: if Agent-TaxOptimizer accesses unauthorized client data, is Agent-Planner or userAlice_DID responsible?

    - An OAuth token is a credential used to access protected resources on behalf of a user or application.
      An OAuth token will be primarily in two forms
    - Access Token: This is a short-lived token that the client uses to access protected resources.
    - Refresh Token: This is a long-lived token used to obtain new access tokens once the initial access token expires.

- **Limited Context Awareness:** Traditional IAM decisions are largely based on static roles or scopes, with minimal understanding of the runtime context, agent intent, or associated risk level. Access is often granted at the beginning of a session and persists, irrespective of evolving circumstances.

    - *Example:* An inventory management agent (Agent-Inventory) has permissions to update stock levels (inventory.write).

        - If it attempts to update stock levels for a product that has been recalled (an environmental condition) or tries to zero out all inventory (anomalous behavior), traditional IAM systems typically lack the contextual awareness to flag this as suspicious or dynamically restrict the permission.

- **Scalability Issues with Token/Session Management:** For organizations deploying hundreds or thousands of (potentially ephemeral) agents, each potentially interacting with numerous services, the volume of authentication events and tokens can overwhelm traditional IAM infrastructure. Managing issuance, validation, and especially revocation of a massive number of short-lived tokens becomes an operational nightmare.

- The volume of tokens that need to be generated and validated can put a significant load on the Identity and Access Management (IAM) infrastructure.

    - *Example:* An e-commerce platform deploys thousands of personalized shopping assistant agents for users.

        - In this example, each agent might exist for only a few minutes.
        - The overhead of frequent, secure token management with traditional protocols is a significant barrier.

- **Dynamic Trust Models & Inter-Agent Authentication:** Agents in MAS often need to authenticate and authorize each other, potentially across organizational boundaries, without a pre-existing, universal trust fabric. OAuth and SAML assume a hierarchical trust model (user trusts IdP, SP trusts IdP). Peer-to-peer trust establishment between autonomous agents from different trust domains is not natively supported.

    - *Example:* Agent-Alpha from "AlphaCorp" needs to request data processing from Agent-Beta from "BetaInc."

        - How do they mutually authenticate?

        - How does Agent-Beta verify Agent-Alpha's capabilities or authorization to request this specific processing without resorting to cumbersome pre-shared secrets or custom API key mechanisms for every pair of interacting agents? The Secure Production Identity Framework for Everyone (SPIFFE) can help Agent-Alpha and Agent-Beta mutually authenticate by providing each with a unique, verifiable identity, eliminating the need for pre-shared secrets or custom API keys between every agent pair. However, SPIFFE only handles authentication—it confirms who each agent is. It does not verify what actions Agent-Alpha is authorized to perform. To check Agent-Alpha's capabilities or permissions for a specific data processing request, Agent-Beta would still need a separate authorization system or policy engine that interprets SPIFFE identities and enforces access control.

- **NHI Proliferation and Management Crisis:** Each autonomous agent may require NHIs for numerous APIs, databases, and services, leading to an exponential growth in secrets that must be securely stored, rotated, and managed. This "secret sprawl" significantly increases the attack surface.
    - *Example:* A single supply chain optimization agent might need API keys for:
        - a shipping provider
        - a warehousing system
        - a customs declaration service
        - an internal ERP

- **Security and Compliance issues :** short-lived tokens enhance security by reducing the window of exploitation, they increase the frequency of token issuance and validation.
- Maintaining logs and audit trails for a large number of tokens is crucial for compliance but can be overwhelming.
- **Global Logout/Revocation Complexity:** If an agent is compromised or its task is complete, ensuring its access rights and sessions are immediately and comprehensively revoked across all systems it interacts with is a major challenge with traditional, often session-based protocols. Fragmented revocation mechanisms can leave lingering access.
    - *Example:* An agent Agent-DataAggregator has active sessions with three different microservices using OAuth tokens.
        - If the agent is detected as compromised, revoking its token at the authorization server is step one.
        - Ensuring each microservice immediately invalidates its session based on that token, especially if they cache permissions, requires a coordinated effort not always inherent in standard OAuth.

## 2.2. Why Agentic AI Demand a New IAM Paradigm Beyond Traditional Protocols

Beyond the protocol mismatches, the very nature of agentic AI introduces further complexities:

- **Autonomy and Potential Unpredictability:** Agents with high degrees of autonomy can make decisions that were not explicitly programmed, potentially leading to unforeseen interactions or resource access attempts that challenge static policy definitions.
- **Ephemerality and Dynamic Lifecycles:** Agents can be created, cloned, and destroyed rapidly based on demand. Managing identities and access for such transient entities with persistent credentials is risky and inefficient. An "ephemeral authentication" approach is needed. Ephemeral authentication could involve time-limited credential or context-aware tokens that are dynamically issued and revoked as agents are initiated and destroyed, minimizing exposure from long-lived credentials. While this challenge isn't unique to AI agents, it becomes significantly more critical due to their autonomous nature and ability to make decisions and take actions in real time, potentially amplifying security breaches before human intervention is possible.
- **Evolving Capabilities and Intent:** Agents, particularly those incorporating online learning, can adapt their behavior and even their goals over time. An IAM system must incorporate adaptive policies to detect agent evolving capabilities diverge from authorized goal, preventing unauthorized privilege escalations.
- **Need for Verifiable Provenance and Accountability:** Tracing actions back to a specific agent instance, understanding its decision-making process (especially if it involved other agents or tools), and ensuring non-repudiation is crucial for trust and forensics.
- **Preventing Autonomous Privilege Escalation:** A sophisticated agent might probe its environment or interact with management APIs to grant itself higher privileges if not carefully constrained. Additionally, agents may interact with each other in a way that leads to privilege escalation through their combined actions, in a manner similar to collusion among humans, necessitating IAM systems that can detect and mitigate such collusive behavior dynamically.
- **Risks of Over-Scoping Access and Permissions**: Agents will actively explore and utilize every permission available to them to perform the task assigned to them. This pervasive behavior demands a shift to tightly scoped, task-specific, and context-based access controls to prevent over-privilege and unintended access to sensitive data and environments. Dynamic least-privilege access model that adjusts permission in real time based on agent tasks and context is essential to prevent over-privilege.
- **Secure and Efficient Cross-Agent Communication & Collaboration:** As agents increasingly form ad-hoc teams or workflows, the need for secure, low-overhead authentication and authorization between them becomes paramount. Mutual TLS, Decentralized Identifiers, Zero-trust micro segmentation to secure inter-agent communication with minimal latency.
- **Actions Taken May Not Directly Correlate to Human Requests**: As agents are given increasing autonomy and reasoning capabilities, the direct tie between a given human goal and

actions taken by any particular agent may no longer exist. For example, a management agent may decide to request a worker agent to use a tool based on its own reasoning, rather than at the specific request of a human. An IAM system must be able to discern between when an action is taken at the direct request of a human, and when it is the result of an agentic decision or a mix of these two.

These challenges collectively demonstrate that a reactive, bolt-on approach to agent IAM is insufficient. A proactive, purpose-built architectural framework is imperative to harness the power of MAS securely and responsibly. Traditional IAM systems provide a shaky foundation for the towering edifice of interconnected, autonomous AI agents.


*Figure 2: Key Limitations of Traditional IAM in Multi-Agent Systems*

# 3. Defining the Agent Identity (Agent ID) in Multi-Agent System

To address the challenges of Agentic AI IAM, we must first redefine what constitutes an "identity" for an AI agent. It transcends a simple API key or a username/password. An Agent ID in a MAS context must be a rich, verifiable, dynamic, and cryptographically secured profile that serves as the foundation for trust, access control, and accountability. This expands the notion of identity to not only support authentication and access but also underpin dynamic authorization, accountability and behavioral governance.

## 3.1. What Constitutes an AI Agent's Identity? Beyond Static Identifiers

An AI agent's identity is not merely a label but a comprehensive digital representation that captures its origin, purpose, capabilities, behavior, relationships, and attestations. Agent IDs represent a subset of NHIs that are autonomous, goal-driven, and context-aware. However, to function effectively, agents also rely on or control other types of NHIs (i.e., API tokens, service accounts, workload identities access external resources, execute API calls, or authenticate to services). Agent IDs must be unique across the system and throughout time, even when agents are cloned or operate ephemerally the claims made by or about the agent can be verifiable. We define an "instance" of an AI agent as a runtime instantiation of an agent's software and model, combined with its unique state, memory, and interaction history at a given point in time. **Table 1** outlines different identity models for agents based on their lifespan, origin, and hierarchical relationships, highlighting how unique identifiers support traceability and attribution.

In addition to origin and attribution, the lifecycle of an Agent ID must be well-defined. Identity termination – whether through task completion, behavioral anomalies or administrative revocation or expiration policies – is as critical as its creation. Without such control, dormant or orphaned Agent IDs risk becoming security liabilities, especially in large-scale MAS systems.

| Agent Type | Description |
|---|---|
| **Persistent Agents** | For long-lived agents, the ID provides a continuous thread of identity across sessions, state changes, and even restarts, as long as core attributes and memory persist. |
| **Ephemeral Agents** | Each execution of a short-lived, task-specific agent constitutes a new instance with a unique (potentially derived) ID, ensuring that its actions are distinctly attributable, even if its lifespan is mere seconds. Thus, the lifecycle of an ephemeral agent includes instantiation, DID assignment, Verifiable Credential issuance, runtime operation, and teardown. |

| | |
|---|---|
| **Agent Copies/Forks** | A copied or forked agent with the same codebase becomes a distinct instance with its own unique ID, diverging from its parent over time. The relationship to the parent (provenance) should be part of its identity. |
| **Hierarchical Agents** | Sub-agents spawned by a parent agent are separate instances, each with a unique ID, but with a verifiable link (e.g., via a Verifiable Credential) back to the parent, enabling traceable delegation. |

*Table 3: Agent Identity Models in Multi-Agent Systems*

## 3.2. Essential Components of an Agent ID:

The proposed Agent ID, ideally anchored by a Decentralized Identifier (DID) (W3C, 2022), should encapsulate a wide array of information within its associated DID Document and through Verifiable Credentials (VCs) (W3C, 2021; Sporny et al., 2024). These components allow for a holistic representation:

**(A) Cryptographic Anchor & Verifier:**

- **Decentralized Identifier (DID):** The globally unique, persistent, and resolvable root identifier (e.g., did:example:agent123). The DID method dictates how it's registered and resolved.
- **Associated Cryptographic Key Pairs:** Public/private key pairs linked to the DID, specified in the verificationMethod section of the DID Document. These are used for signing agent actions, encrypting communications, and authenticating the agent when it presents its DID.
- **DID Document Service Endpoints:** Pointers to services associated with the agent, such as its communication endpoints or a profile service.

**(B) Core Attributes & Metadata (Often in DID Document or VCs):**

- **Creator/Deployer/Owner/Controller:** DIDs or other identifiers of the entities responsible for the agent's creation, operation, and governance.
- **Agent Software Version & Model Information:** Cryptographic hash of the agent's core model parameters and software version.We recommend the use of FIPS-approved SHA-3 family hash functions (SHA3-224, SHA3-256, SHA3-384, and SHA3-512) to ensure strong cryptographic security.
- **Timestamps:** Creation date, last update, expected expiry (for ephemeral IDs).
- **Dependencies (Optional):** A list of critical software components, libraries, or other agent services that this agent relies upon. This is optional metadata and a normative reference to AIBOM is the preferred way to define the dependencies.
- **Training Information (Optional):** Details about the datasets, methods, and environment used to train the agent's underlying model.
- **Lifecycle Status:** Current state (e.g., active, suspended, revoked, archived).

**(C) Capabilities, Scope, and Behavior (Crucial for Access Control & Trust):**

- **Formal Scope of Behavior:** A machine-readable definition of the agent's intended tasks, operational domains, and interaction boundaries.
- **Decision-Making Capabilities:** Details on the agent's model type, primary reasoning methods, and key behavioral parameters.
- **Toolset:** An explicit, verifiable list of the tools, APIs, or other agents it is authorized to use.
- **Expected Outcomes & Limitations:** Definition of intended successful outcomes and known failure modes or limitations.

**(D) Operational & Security Parameters:**

- **Communication Protocols Supported:** Specification of protocols the agent can use.
- **Security Properties Attested:** Claims about security features.
- **Compliance Information:** VCs asserting compliance with relevant regulations.
- **Update Mechanism:** Information on how the agent's software, model, or DID Document can be securely updated.

**(E) Verifiable Credentials (VCs): The Key to Dynamic Attributes and Trust:** VCs are digitally signed attestations about an agent, issued by a trusted entity. Usually, trusted entities are government agencies or big IT companies acting as Certification Authorities. Agents can hold and present these VCs to prove specific attributes or authorizations.

- **Role VCs:** "DisasterResponseCoordinatorRole".
- **Capability VCs:** "CertifiedToUse_MedicalImagingAI_v3".
- **Reputation VCs:** "TrustedCollaborator_Score_95_Percentile_from_CommunityX".
- **Provenance VCs:** "SpawnedBy_did:example:parentAgent789_at_TimestampZ".

## 3.3. Agent ID Ownership and Control:

A cornerstone of this new IAM paradigm is the principle of Self-Sovereign Identity (SSI) applied to agents.

- **Agent (or its designated controller) as Holder:** The agent itself or its designated controller holds the private keys associated with its DID and manages its VCs.
- **Controller:** The entity ultimately responsible for the agent.
- **Decoupling from Issuers and Verifiers:** The agent's identity is not solely dependent on a single centralized identity provider.

This model moves away from centrally managed identities, empowering the agent/controller with greater control and portability.

## 3.4. ID Generation, Assignment, and Lifecycle Management: From Birth to Revocation

Managing the lifecycle of these rich Agent IDs is crucial.

- **Initial ID Generation and Assignment using different approaches:**
    - **Centralized Platform Issuance:** In enterprise settings, a platform might generate a DID for an agent upon deployment.
    - **Decentralized/Self-Issuance:** An agent or its controller can generate its own DID using a suitable DID method

At creation, the DID can be associated with core attributes(See Section 3.2B).

- **Runtime ID Adaptation & Ephemeral Identities:** Agents may need to operate under different personas or with limited-scope identities for specific tasks.
    - **Role-Based/Task-Specific IDs:** An agent might present a specific VC that grants it a temporary role or use a derived, short-lived DID.
    - **Secure Protocol for Assuming Runtime IDs:**
        1. **Request:** The agent requests a new role/ephemeral ID/VC.
        2. **Verification:** Issuer verifies primary DID and policies.
        3. **Issuance:** Issuer provides a new (potentially time-bound, scope-limited) VC or ephemeral DID.
        4. **Usage:** Agent uses the new ID/VC for the specific context.
        5. **Revocation/Expiry:** The temporary ID/VC is revoked or expires.

- **ID Update and Revocation:**
    - **DID Document Updates:** Changes to an agent's capabilities or keys require updating its DID Document.
    - **VC Revocation:** Invalid VCs must be revoked using mechanisms like VC Status Lists.
    - **DID Deactivation/Revocation:** The primary DID can be marked as deactivated if the agent is decommissioned.

This rich, dynamic, and verifiable Agent ID serves as the cornerstone of the proposed Agentic AI IAM framework. The demo SDK for Agent ID is published as open source code on Github (Huang, 2025c).

**AIAM** enables secure identity, intent, and access control for autonomous agents operating within distributed systems.

It supports fine-grained delegation, context-aware authorization, and real-time trust evaluation across multi-agent workflows. Reference of this AIAM SDK and the Demo video of this security Functionality. (Sheriff,2025d)

**In conclusion**, the above establishes the need for a more granular, cryptographically verifiable identity model that goes beyond the conventional IAM. Traditional models lack the mechanisms to accommodate ephemeral agents, decentralized trust anchors, and continuous validation workflows. The following section on the framework architecture is motivated by the design decisions made above.

# 4. The New Agentic AI Identity and Access Management Framework Architecture

## A. Zero-Trust Architecture for Agentic AI

Core Principles:

- Never trust, always verify: Every agent action requires real-time verification regardless of previous authentication
- Assume breach: Design systems expecting that agents may be compromised or manipulated
- Least privilege: Grant agents only the minimum access required for their specific task

Implementation Strategy:

- Continuous verification of agent identity and behavior through behavioral analysis and anomaly detection
- Micro-segmentation of AI environments to limit lateral movement if agents are compromised
- Dynamic policy enforcement that adapts to real-time risk assessments

## B. Decentralized Identity Management

Decentralized Identifiers (DIDs) and Verifiable Credentials:

Modern agentic AI systems require decentralized identity frameworks that provide:

- Self-sovereign identity for AI agents, allowing them to maintain their own identity without relying on centralized authorities
- Verifiable credentials that can be cryptographically verified without contacting the issuing authority
- Selective disclosure capabilities using zero-knowledge proofs to share only necessary information

Agent Identity Architecture:

- Rich agent identities that encapsulate capabilities, provenance, behavioral scope, and security posture
- Agent Naming Service (ANS) for secure, capability-aware discovery of agent services
- Cryptographic attestation of agent integrity and authenticity

## C. Dynamic Policy-Based Access Control (PBAC)

Policy Engine Architecture:

Replace static role-based controls with dynamic, context-aware policy engines that:

- Evaluate access requests in real-time based on multiple attributes including agent identity, resource sensitivity, environmental context, and risk factors
- Support natural language policy definitions that can be translated into executable access control rules
- Implement fine-grained resource filtering both before and after data retrieval

Key Components:

- Attribute-based evaluation considering user identity, agent capabilities, resource classification, and environmental factors
- Intent-based authorization that understands and evaluates the purpose behind access requests
- Dynamic policy updates that adapt to changing threat landscapes and business requirements

## D. Authenticated Delegation Framework

Secure Delegation Mechanisms:

Implement OAuth 2.1 extensions specifically designed for AI agents:

- Agent-specific credentials that clearly distinguish between human users and AI agents
- Delegated authorization flows that maintain clear chains of accountability from human principals to agents
- Scoped permission models that allow fine-grained control over what agents can access and modify
- Revocable delegation enabling real-time termination of agent permissions

Implementation Elements:

- requested_actor parameter in authorization flows to specify which agent is requesting delegation
- actor_token parameter for agent self-authentication during token exchange
- Enhanced token claims capturing the complete delegation chain for audit and accountability

## E. Continuous Monitoring and Behavioral Analytics

Real-Time Monitoring:

- Agent behavior tracking to detect deviations from expected patterns
- Anomaly detection using machine learning to identify potentially compromised agents
- Audit trail generation that captures all agent actions with full context and authorization basis
- Performance monitoring to detect resource abuse or denial-of-service attempts

Trust Scoring:

- Dynamic trust scores based on agent behavior, historical performance, and security posture
- Continuous risk assessment that adjusts agent permissions based on current threat levels
- Automated response mechanisms that can restrict or terminate agent access when anomalies are detected

## F. Secure Communication Protocols

Enhanced MCP Security:

- Cryptographic server verification to ensure clients connect to legitimate MCP servers
- Enhanced Tool Definition Interface (ETDI) with cryptographic identity verification and immutable versioned tool definitions
- Policy-based access control for MCP tool interactions, evaluating capabilities against explicit policies using dedicated policy engines

Agent-to-Agent Communication:

- Secure A2A protocol implementation with enterprise-grade authentication and authorization
- Multi-modal support with proper access controls for different communication types
- Task-oriented security that maintains security boundaries throughout long-running collaborative tasks

To address the multifaceted challenges of managing AI agents in MAS, we propose a comprehensive IAM framework built upon modern cryptographic primitives and a layered architecture designed for dynamic, secure, and interoperable agent interactions.

## 4.1. Foundational Pillars

The framework rests on several key technological pillars:

- **A. Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs):**
    - DIDs (W3C, 2022) provide globally unique, persistent, cryptographically verifiable identifiers controlled by the agent or its controller, enabling self-sovereign identity essential for cross-organizational and decentralized MAS. Each agent instance receives a unique DID, providing a persistent identity anchor throughout its operational lifecycle. VCs (W3C, 2021; Sporny et al., 2024) are digitally signed attestations about an agent, allowing granular and dynamic proof of attributes, capabilities, or authorizations, and even metadata such as validity periods and credential status references.These technologies are particularly well-suited for representing Non-Human Identities (NHIs), which are widely discussed in the industry (OWASP, 2025; Cloud Security Alliance, 2024), providing a standardized approach to managing autonomous agent identities in distributed systems.
- **B. Zero-Knowledge Proofs (ZKPs):**
    - ZKPs (Goldwasser et al., 1989) allow an agent to prove a statement's truth (e.g., possessing a specific VC attribute) without revealing the underlying information, balancing verifiability with privacy. This is crucial for selective disclosure and proving policy compliance without exposing sensitive internal states.
- **C. Agent Naming and Discovery Service (ANS):**
    - An ANS, inspired by DNS but tailored for agents, enables secure and reliable discovery based on capabilities, protocols, providers, and versions, not just names(Huang, Narajala, Habler, & Sheriff, 2025). This could use a naming structure like protocol://AgentFunction.CapabilityDomain.Provider.Version[.protocolExtension] and resolve to DIDs, with entries secured by PKI or linked to verifiable claims.

## 4.2. Core Architectural Layers

The proposed framework is structured in layers (Figure 1):


*Figure 3: Agentic IAM Core Architectural Layers*

**(Layer 1) Identity & Credential Management Layer:** Responsible for creating, issuing, storing, and managing the lifecycle of Agent DIDs and VCs.

- **DID Registries/Methods:** Systems anchoring DIDs and their DID Documents (e.g., public/permissioned DLTs, did:web, an "Agent ID Provider Network").
- **VC Issuers and Verifiers:** Trusted entities issuing and checking VCs.
- **Agent Wallets/Secure Storage:** Secure agent-side storage for private keys and VCs.
- **Key Management Services:** For key generation, rotation, and revocation.

**(Layer 2) Agent Discovery and Trust Establishment Layer:** Enables agents to find each other and establish trust.

- **ANS Resolution Mechanisms:** Services implementing the ANS for capability-based discovery.
- **DID Resolvers:** Standard components for retrieving DID Documents.

- **Reputation Systems:** DID-anchored systems for sharing reputation scores.
- **Trust Frameworks:** Policies defining how trust is evaluated (e.g., trusted VC issuers).

**(Layer 3) Dynamic Access Control Layer:** Makes fine-grained, context-aware authorization decisions.

- **Policy Decision Point (PDP):** Evaluates access requests against policies using agent ID (DID, VCs), resource attributes, action, and context.
- **Policy Administration Point (PAP):** Where policies (e.g., in Rego/OPA) are defined.
- **Policy Information Point (PIP):** Gathers attributes for the PDP.
- **Access Control Mechanisms:** ABAC, PBAC, and JIT access using temporary, scoped VCs.

**(Layer 4) Unified Global Session Management & Policy Enforcement Layer:** A critical innovation for consistent, real-time establishment, tracking, management, and enforcement of IAM policies, including global logout and session invalidation, across heterogeneous agent communication protocols.

- **Cross-Protocol Session Authority (SA):** Logically centralized component for global session oversight, policy distribution, orchestrating global logout, and state change propagation.
- **Adapter Enforcement Middleware (AEM):** Lightweight plugins injected into Protocol Adapters, hooking into session initiation, subscribing to SA updates (via SSS), intercepting requests, and enforcing decisions locally, including terminating local sessions on global logout.
- **Enhanced Protocol Adapters:** Gateways understanding specific agent protocols, integrated with AEM for authentication, authorization, and local session management linked to global contexts.
- **Session State Synchronizer (SSS):** Highly available, low-latency distributed data store maintaining a real-time ledger of active global agent session contexts, their mappings to protocol-specific sessions, and current validated capabilities/status. It's the primary source of truth for AEMs regarding session validity. As an alternative to the distributed ledger - the OpenID Shared Signals Frameworks can be used for events across entities and the systems subscribed to the events or have publishing events will help achieve this requirement. It can help expedite their adoption rate based on their SSF implementations.

*Flow Example: Global Logout for Agent Alpha*

1. Global logout for AgentAlpha_DID reaches SA.
2. SA updates SSS: marks GlobalSessionID_123 (for AgentAlpha_DID) as "terminated".
3. SA may push notifications to relevant AEMs.
4. AEM for A2A adapter, on SSS check (or push), sees termination, invalidates local A2A session.
5. Similar for MCP adapter's AEM. Further requests from Agent Alpha are blocked.

## 4.3. Applying Zero Trust Principles

The framework embodies Zero Trust (Kindervag, 2010):

- **Explicit Verification:** Always verify agent identity (DID, VCs) and authorization.
- **Least Privilege Access:** Grant minimum necessary permissions, ideally via JIT VCs. Just-in-time verifiable credentials are dynamically issued with specific permissions for limited time periods and particular tasks, automatically expiring when the agent completes its designated function or when the time window closes.
- **Assume Breach:** Design for compromise; rapid revocation via the Unified Enforcement Layer is key.
- **Micro-segmentation:** Granular agent DIDs support network/application micro-segmentation.
- **Data-Centric Security:** Policies tied to data sensitivity and agent capabilities.


*Figure 4: Zero Trust IAM Framework for Multi-Agent Systems*

This architectural foundation establishes a robust lifecycle model for agent identity and verifiable credentials. By clearly delineating ephemeral and persistent identity domains, the framework ensures a minimal attack surface. Subsequent sections will build on this foundation to explore operational monitoring and compliance considerations.

# 5. Agent IDs in the IAM Process

This section provides an in-depth exploration of how these constructs enable robust fine-grained access control, ensure secure and non-reputable logging, facilitate effective real-time monitoring and anomaly detection, and empower agile, targeted incident response. A critical enabler for many of these use cases is the **Agent Name Service** (ANS by Huang, Narajala, Habler, & Sheriff, 2025), which provides a secure and capability-aware mechanism for agents to discover each other before interaction. We will illustrate conceptual design patterns, including sample interactions involving emerging agent communication protocols like Google's Agent-to-Agent (A2A) protocol and Anthropic's Model Context Protocol (MCP), demonstrating the framework's adaptability and practical utility in complex Multi-Agent Systems (MAS).

## 5.1. Fine-Grained Access Control in Action

Effective access control in MAS must move beyond static roles to embrace dynamic, attribute-based , and policy-driven methodologies. The journey often begins with an agent needing to discover another agent or service capable of fulfilling a specific need. This is where the ANS plays a pivotal role, integrated with DIDs and VCs for subsequent secure interaction and authorization.

- **Deep Dive into Dynamic Authorization Decisions, Prefaced by ANS Discovery:** Consider TaskOrchestratorAgent (did:com:enterprise:agent:orchestrator:alpha-001) which needs to delegate a financial data analysis task. Its first step is to find a suitable agent. It queries the Agent Name Service (ANS) for an agent that matches certain criteria.

  **1. ANS Discovery Phase:** TaskOrchestratorAgent constructs an ANS query. The ANS is designed for capability-aware resolution, using a structured naming convention such as: Protocol://AgentID.agentCapability.Provider.vVersion.Extension.

```
Conceptual ANS Query (e.g., via a secure API call to an ANS resolver):

// Request to ANS Resolver

{

 "requestType": "resolveAgentByCapability",

 "desiredProtocol": "acp", // Prefers Agent Communication Protocol

 "requiredCapability": "FinancialRiskAnalysis.CorporateReporting",
```

```
 "preferredProvider": "AcmeFinanceServices",
 "versionRange": ">=2.1.0 <3.0.0", // Semantic versioning for the agent's capability
 "requiredAttestations": [ // Optional: request agents with specific VCs
  { "vcType": "SOXComplianceCertified" }
 ]
}
```

The ANS resolver (itself a secure, trusted component of the IAM framework, potentially with its own DID and verifiable responses) queries its Agent Registry. The Agent Registry stores information about registered agents, including their ANSNames, DIDs, PKI certificates (if using a PKI-centric ANS as described in this paper), and protocolExtensions detailing their capabilities and associated VCs.

```
Conceptual ANS Resolution Response:
// Response from ANS Resolver
{
 "resolutionStatus": "success",
 "resolvedAgents": [
  {
   "ansName":
"acp://RiskAnalyzerBot.FinancialRiskAnalysis.AcmeFinanceServices.v2.1.3.prod",
   "agentDid": "did:com:acme:agent:riskanalyzer:beta-007",
   "serviceEndpoint": "acps://riskanalyzer.acmefinance.com/service",
   "protocolExtensions": {
    "acp": { "supportedMessagePatterns": ["request-response", "publish-subscribe"] }
   },
   "relevantVcSnippets": [ // Snippets or pointers to VCs that matched query
```

```
      { "type": "SOXComplianceCertified", "issuer": "did:com:acme:audit:sox-issuer",
"issueDate": "2025-01-15" }

    ],

    "ansRecordSignature": "..." // Signature by the ANS resolver over this record

  }

  // Potentially other matching agents

 ]

}
```

TaskOrchestratorAgent verifies the ansRecordSignature. It now has the DID of a candidate: RiskAnalyzerBot (did:com:acme:agent:riskanalyzer:beta-007).

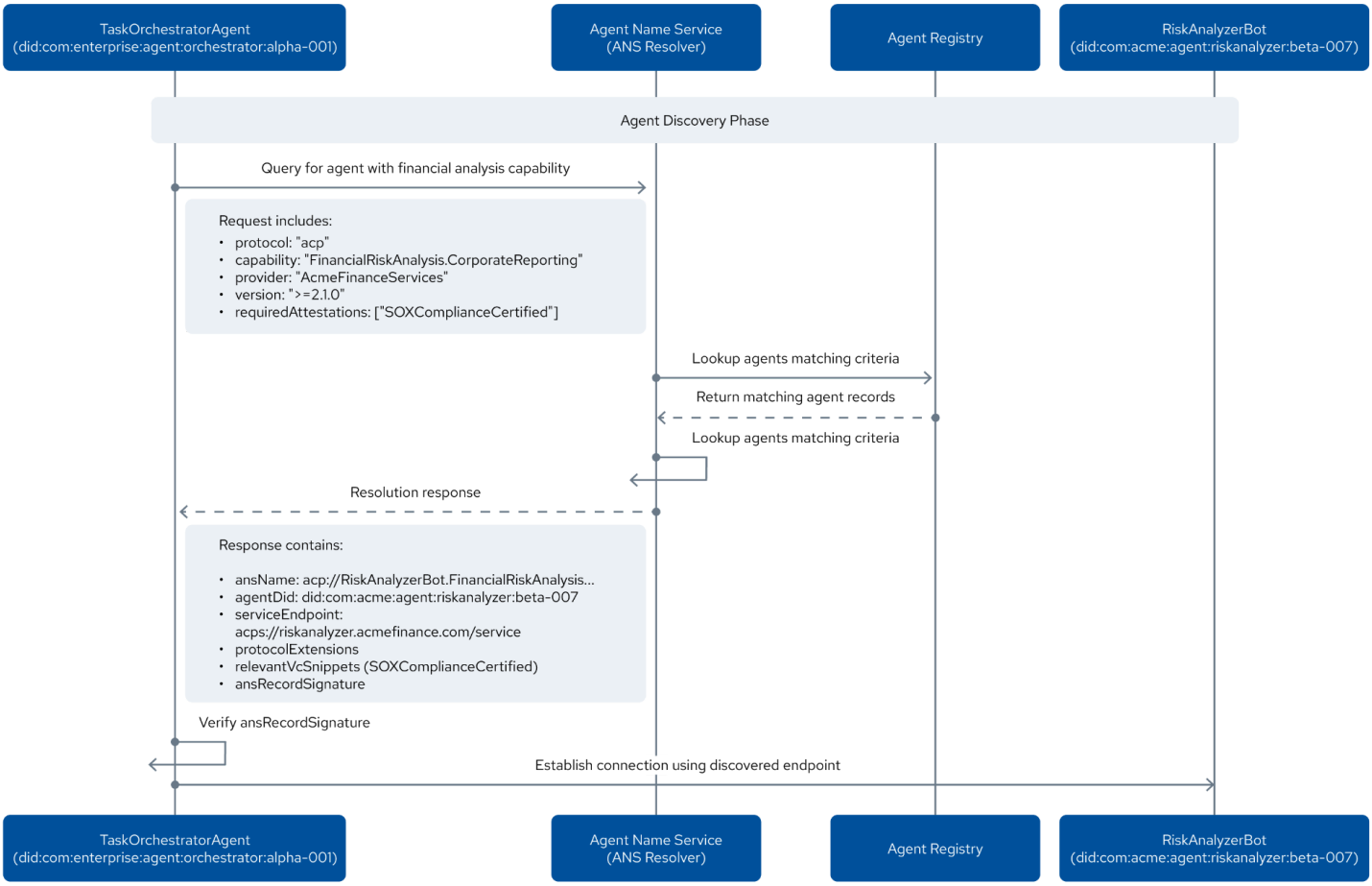


*Figure 5: Agent Discovery and Registration Protocol Flow*

**2. Interaction and Dynamic Authorization:** TaskOrchestratorAgent now initiates communication with RiskAnalyzerBot (e.g., via ACP). As part of establishing this secure channel or with its first request, RiskAnalyzerBot needs to access InternalDB-SalesFigures and ExternalAPI-MarketSentiment.

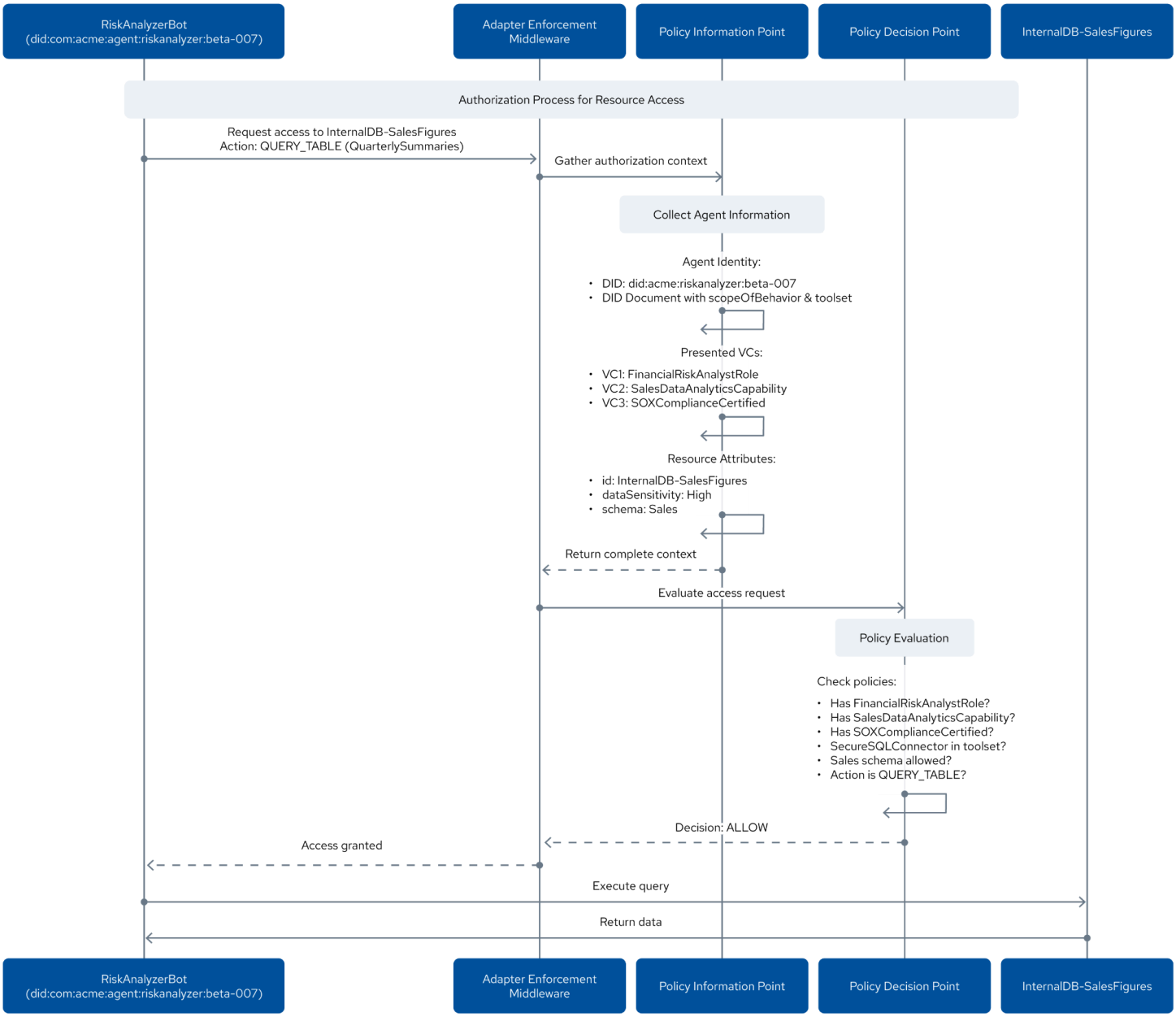


*Figure 6: Dynamic Authorization Flow with Adapter Enforcement Middleware*

The request from RiskAnalyzerBot (let's call it did:acme:riskanalyzer:beta-007) to access InternalDB-SalesFigures is intercepted by the **Adapter Enforcement Middleware (AEM, see section 4)**. The AEM/PIP gathers:

- **Agent Identity:**
  - RiskAnalyzerBot's DID: did:acme:riskanalyzer:beta-007.
  - Its resolved DID Document might state: scopeOfBehavior: "Perform financial risk analysis based on sales and market data." toolset: {"toolName": "SecureSQLConnector", "targetSchemas": ["Sales", "Projections"]}.

- **Presented VCs (obtained during its registration or dynamically):**
  - VC1 (Role): { "type": "FinancialRiskAnalystRole", "issuer": "did:com:acme:hr", ... }
  - VC2 (Capability): { "type": "SalesDataAnalyticsCapability", "issuer": "did:com:acme:datascience", ... }
  - VC3 (SOX Compliance - discovered via ANS): { "type": "SOXComplianceCertified", "issuer": "did:com:acme:audit:sox-issuer", ... }
- **Resource Attributes:** id: InternalDB-SalesFigures, dataSensitivity: High.
- **Action:** QUERY_TABLE (QuarterlySummaries).
- **Context:** requestTime, sourceIpSegment.

The PDP evaluates this against policies. For example:

```
package acme.data_access

default allow = false

# Allow access if agent has correct role, capability VCs, SOX compliance,

# and the requested action is within its declared toolset capabilities for the resource.

allow {

  input.agent.vcs[_].credentialSubject.role == "FinancialRiskAnalystRole"

  input.agent.vcs[_].credentialSubject.capability == "SalesDataAnalyticsCapability"

  input.agent.vcs[_].type[_] == "SOXComplianceCertified" // Check for presence of type



  # Verify toolset from resolved DID Document (assuming toolset populated by PIP)

  some tool_idx

  allowed_tool := input.agent.did_document.service[_].serviceEndpoint.toolset[tool_idx]

  allowed_tool.toolName == "SecureSQLConnector"

  input.resource.schema IN allowed_tool.targetSchemas // e.g., "Sales"

  input.resource.id == "InternalDB-SalesFigures"
```

```
  input.action == "QUERY_TABLE"

  input.resource.table == "QuarterlySummaries" // More granular check

}
```

The ANS discovery step ensures that TaskOrchestratorAgent doesn't just *find* an agent, but finds one that *verifiably claims* relevant capabilities and compliance (like SOXComplianceCertified) *before* even attempting interaction. The subsequent authorization then re-verifies these claims (via presented VCs) and checks against more granular policies for resource access. This two-step process (secure discovery then secure, fine-grained authorization) is crucial for building trust and efficiency in large MAS. The DID is the consistent thread linking the discovered entity in ANS to the entity being authorized.

- **Just-In-Time (JIT) Access, Enhanced by ANS for Tool Discovery:** Imagine DataProcessingAgent-Temp77 (did:ephemeral:task-xyz:agent-77) is a short-lived agent spawned by WorkflowEngine to perform a specific data transformation. It needs temporary access to a specialized DataTransformationTool-Q.

    1. **ANS for Tool Discovery:** WorkflowEngine (or DataProcessingAgent-Temp77 itself if it has this capability) first queries the ANS to discover a suitable and currently available instance of DataTransformationTool-Q. *ANS Query:*

```
{
 "requestType": "resolveAgentByNameAndCapability",
 "ansNamePattern": "mcp://DataTransformationTool-Q.*.AcmeTools.v1.*.internal",
 // Using wildcard for AgentID part if multiple instances exist
 "requiredCapability": "VectorEmbeddings.HighDimReduction",
 "availabilityRequirement": "online_accepting_jobs" // Custom ANS extension
}
```

    The ANS returns the DID of an available instance, e.g., did:com:acmetools:mcp:tool:transformQ:instance03.

    2. **JIT VC Issuance via MCP Context (Conceptual):** WorkflowEngine (acting as a trusted issuer for this context) issues a JIT VC to DataProcessingAgent-Temp77:

```
{
```

```
  "type": ["VerifiableCredential", "MCPToolAccessPass"],
  "issuer": "did:com:acme:workflow:engine-issuer",
  "validFrom": "2025-10-02T14:30:00Z",
  "validUntil": "2025-10-02T14:45:00Z", // Valid for 15 mins
  "credentialSubject": {
   "id": "did:ephemeral:task-xyz:agent-77",
   "authorizedToolDID": "did:com:acmetools:mcp:tool:transformQ:instance03",
   "allowedActions": ["executeTransform"],
   "inputDataHandle": "blob://temp-input-xyz",
   "outputDataHandle": "blob://temp-output-xyz",
   "jobId": "job-ephemeral-77a"
  }
}
```

3. **MCP Tool Invocation with JIT VC:** DataProcessingAgent-Temp77 invokes DataTransformationTool-Q (whose MCP endpoint was found via ANS then DID resolution). It presents this JIT VC within the MCP call. The following figure is a high level overview.

   *Conceptual MCP Call (e.g., using gRPC or HTTP, carrying VC in metadata/headers):* Let's assume MCP uses gRPC and metadata for auth as customized transport.

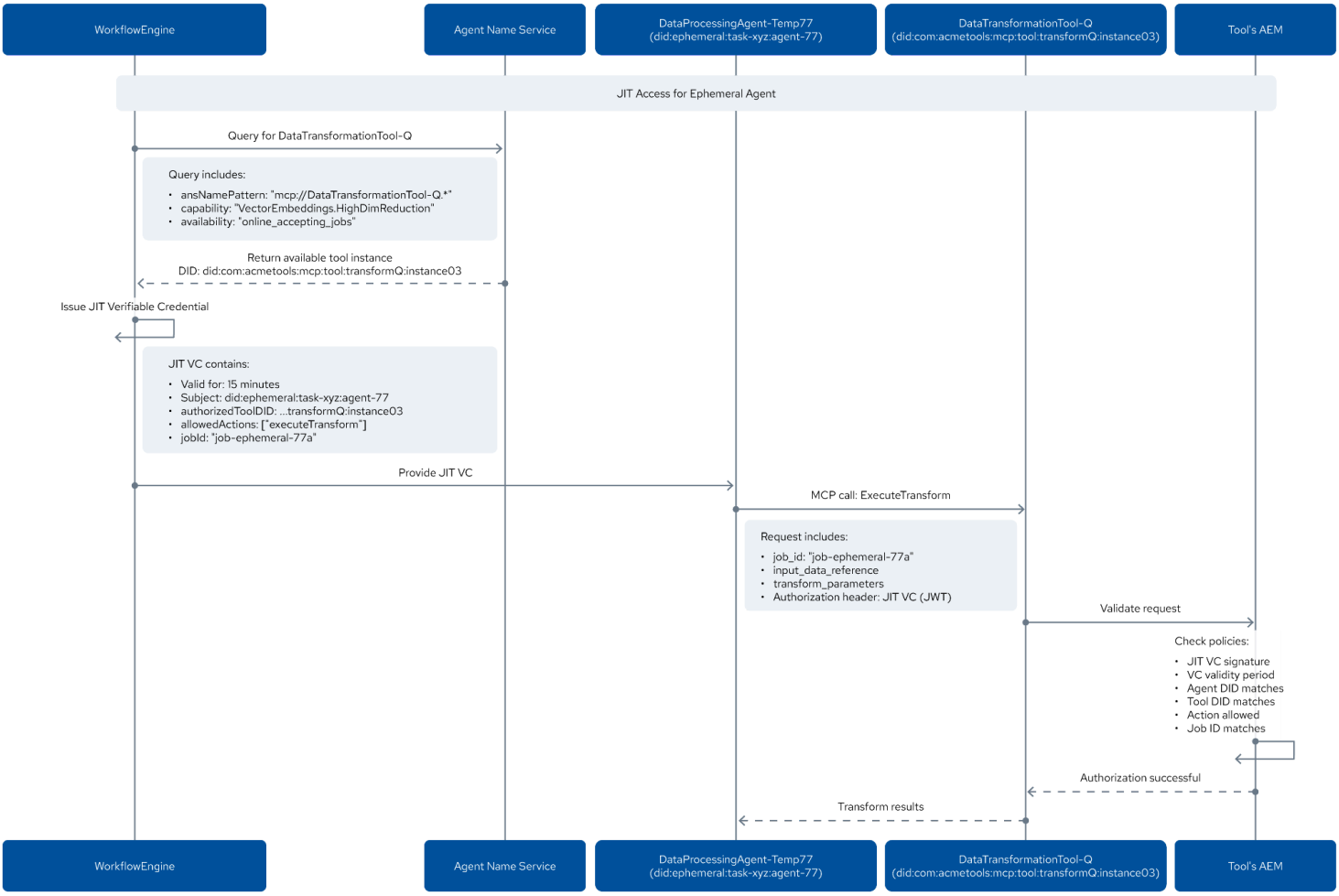


*Figure 7: Just-In-Time Verifiable Credential Issuance for Ephemeral Agents*

```
// Conceptual .proto definition for an MCP tool call

service TransformationTool {

 rpc ExecuteTransform(TransformRequest) returns (TransformResponse);

}

message TransformRequest {

 string job_id = 1;

 string input_data_reference = 2; // "blob://temp-input-xyz"

 map<string, string> transform_parameters = 3;
```

```
}

Client-side pseudocode for DataProcessingAgent-Temp77:

# Assume 'mcp_tool_stub' is the gRPC stub for DataTransformationTool-Q

# Assume 'jit_vc_jwt' is the JIT VC serialized as a JWT

metadata = [

  ('x-agent-did', 'did:ephemeral:task-xyz:agent-77'),

  ('authorization-vc', jit_vc_jwt)

] # gRPC metadata

request_payload = TransformRequest(

  job_id="job-ephemeral-77a",

  input_data_reference="blob://temp-input-xyz",

  transform_parameters={"algorithm": "PCA", "dimensions": 128}

)

try:

  response = mcp_tool_stub.ExecuteTransform(request_payload,
metadata=metadata)

  # Process response and write to "blob://temp-output-xyz"

except grpc.RpcError as e:

  # Handle authorization failure or tool error

  log(f"MCP tool call failed: {e.details()}")
```

4. **Verification at MCP Tool's AEM:** The AEM for DataTransformationTool-Q extracts and verifies the DID and jit_vc_jwt. The PDP checks if did:ephemeral:task-xyz:agent-77 is authorized by this specific VC to call *this tool instance* (did:com:acmetools:mcp:tool:transformQ:instance03) for executeTransform with the given jobId and data handles, and if the VC is within its validity period.

ANS helps find the *right instance* of a potentially multi-instance MCP tool. The JIT VC then provides extremely narrow, time-bound permission *for that specific job and data*, dramatically reducing risk compared to the ephemeral agent having broader, longer-lived credentials for a generic tool type.

- **Capability-Driven Authorization with A2A Protocol:** AlertingAgent-SystemX (did:com:sysX:a2a:alerter:main:v1) needs to send a critical security alert to a SOCDashboardAgent-PlatformY (did:com:platY:a2a:socdash:primary:v2).

    1. **ANS Discovery:** AlertingAgent-SystemX resolves a2a://SOCDashboardAgent.SecurityAlertIngestion.PlatformY.v2.critical via ANS to find the DID and A2A endpoint of SOCDashboardAgent-PlatformY. The ANS response might also indicate that the SOC agent requires alerts to be signed with a key whose DID is on an approved list.

    2. **A2A Message Construction with IAM Context:** AlertingAgent-SystemX holds a VC: {"type": "CriticalAlertSourceCredential", "issuer": "did:com:sysX:security-authority", "credentialSubject": {"id": "did:com:sysX:a2a:alerter:main:v1", "authorizedAlertTypes": ["SECURITY_CRITICAL", "SYSTEM_DOWN"]}}.

        *Conceptual A2A Message from AlertingAgent-SystemX (JSON-like payload for an A2A message):*

```
{
 "a2aHeader": { // Fields defined by A2A spec
  "messageId": "msg-uuid-9876",
  "senderId": "did:com:sysX:a2a:alerter:main:v1", // Using DID as A2A ID
  "recipientId": "did:com:platY:a2a:socdash:primary:v2",
  "protocolVersion": "A2A/1.0"
 },
 "iamExtension": { // Custom extension for our IAM framework
  "verifiablePresentation": [ /* JWT of CriticalAlertSourceCredential */ ],
  "messageSignature": { // Signature over 'a2aHeader' and 'payload'
   "keyId": "did:com:sysX:a2a:alerter:main:v1#key-1", // Key used for signing
   "algorithm": "EdDSA",
```

```
    "signatureValue": "..."
   }
  },
  "payload": {
   "alertType": "SECURITY_CRITICAL",
   "sourceSystem": "SystemX_Firewall_Cluster",
   "details": "Multiple intrusion attempts detected from IP range Z.Z.Z.Z",
   "severity": 5, // 1-5 scale
   "timestamp": "2025-10-02T15:00:10Z"
  }
}
```

Many emerging A2A protocols are defining ways to carry security contexts, often leveraging JWTs or similar token formats within their headers or as part of the message envelope. The iamExtension is a way our framework's specific needs (DID, VP) can be mapped.

3. **Processing at SOCDashboardAgent-PlatformY's AEM:**

    - AEM verifies messageSignature using the public key from did:com:sysX:a2a:alerter:main:v1#key-1 (resolved via DID document).
    - AEM verifies the verifiablePresentation containing the CriticalAlertSourceCredential.
    - PDP checks policies like: "Accept SECURITY_CRITICAL alert IF sender DID holds valid CriticalAlertSourceCredential AND the alert's declared sourceSystem is within the scope covered by that credential."

The ANS ensures AlertingAgent-SystemX reliably finds the authentic SOCDashboardAgent-PlatformY (not an imposter). The VC presented proves the sender is authorized to issue critical alerts, and the message signature ensures integrity and non-repudiation for the alert content. This provides much stronger guarantees than simple IP whitelisting or pre-shared API keys between agents for A2A communication.

The use of ANS for initial discovery, followed by DID-based authentication and VC-based authorization at the point of interaction, forms a robust sequence for secure and fine-grained access control in diverse MAS scenarios.

In addition to validating agent credentials, MAS environments must track role continuity across interaction chains to prevent context hijacks. For instance, agents may attempt to impersonate privileged roles mid-session or nest conflicting prompts within a conversation. Session-aware validation mechanisms such as persistent role tokens, inter-message trust chaining, and nested intent verification can be enforced through middleware or the PDP layer to detect such anomalies. This ensures that agents cannot escalate privileges or override prior authorizations through prompt manipulation

## 5.2. Secure Logging, Auditing, and Non-Repudiation

In systems where autonomous agents perform significant actions, establishing a clear, trustworthy, and irrefutable record of events is paramount. This section delves into how the proposed IAM framework, leveraging rich Agent IDs (DIDs and VCs) and the Agent Name Service (ANS) for discoverable context, transforms logging into a critical component of system integrity, accountability, and auditability.

- **Immutable Agent Identifiers (DIDs) as the Linchpin of Audit Logs:** Every significant action initiated or participated in by an agent MUST be logged with its unique, persistent Decentralized Identifier (DID) as the primary subject identifier. This creates an unambiguous, globally unique, and cryptographically verifiable link to the specific agent instance responsible for any given event.
    - **Enhanced Log Granularity with DID and VC Context:** Beyond simply logging the agent's DID, comprehensive logs should capture:
        - **Precise Timestamp:** Synchronized across the MAS to ensure correct event sequencing.
        - **Agent DID and ANSName:** Logging both the DID (for cryptographic verifiability) and the resolved ANSName (e.g., acp://RiskAnalyzerBot.FinancialRiskAnalysis.AcmeFinanceServices.v2.1.3.prod) provides human-readable context about the agent's role and origin.
        - **Target Resource(s) DIDs/ANSNames:** If the interaction target is another agent or a resource registered in ANS, its DID and ANSName should also be logged.
        - **Request Context Hash:** A canonical SHA-256 hash computed over a deterministic serialization of the request context—including the timestamp, agent DID, target resource identifiers, action type, and normalized input parameters. This hash acts as a unique fingerprint of the request, enabling auditors to validate the integrity of the event without exposing sensitive payload details.
        - **Specific Verifiable Credentials (VCs) Presented:** The unique identifiers (e.g., id or transaction_id) of all VCs presented by the agent to authorize that specific action. For example, logging vc:jwt:uri:issuer-finance-bob:task-q3report2025-instance-002 allows an auditor to later retrieve and verify this exact VC.
        - **DIDs and ANSNames of Collaborating Agents:** In multi-agent tasks, the DIDs/ANSNames of all significant contributing agents should be logged to trace collaborative decision-making.

- **Outcome and Policy Reference:** The result of the action and a reference to the specific policy version (e.g., ACME_Finance_Policy_v3.2.1_Rule7) that permitted it.

- *Example Enriched Log Entry incorporating ANSNames:*

```
{
 "eventId": "evt_20251002T110530Z_A789F123",
 "timestamp": "2025-10-02T11:05:30.123Z",
 "initiatingSystem": "WorkflowOrchestratorInternal", // System originating the top-level task
 "agentDid": "did:com:acme:agent:riskanalyzer:beta-007",
 "agentAnsName": "acp://RiskAnalyzerBot.FinancialRiskAnalysis.AcmeFinanceServices.v2.1.3.prod",
 "actionPerformed": "ExecuteSecureSQLQuery",
 "targetResourceDid": "did:com:acme:resource:db:InternalDB-SalesFigures",
 "targetResourceAnsName": "db://InternalDBSales.FinancialData.AcmeInternal.v1.prod", // If databases are also in ANS
 "inputParametersHash": "sha256-c4d5e6f...",
 "presentedVcIds": [
  "vc:jwt:uri:acme-hr:role-finanalystL2-inst-001",
  "vc:jwt:uri:acme-audit:sox-compliance-inst-003"
 ],
 "decisionPolicyId": "ACME_DataAccess_Policy_v1.7_Rule12b",
 "collaborationContext": {
  "triggeringAgentDid": "did:com:enterprise:agent:orchestrator:alpha-001",
  "triggeringAgentAnsName": "acp://TaskOrchestrator.CoreBusinessLogic.AcmeEnterprise.v1.0.main",
```

```
  "taskId": "task_QuarterlyRiskAssessment_2025Q3"

 },

 "outcome": { "status": "Success", "rowsAffected": 0, "dataRetrievedHash":
"sha256-g7h8i9j..." },

 "logEntrySignature": "..." // Digitally signed by the logging service or the agent
performing the action

}
```

Logging ANSNames alongside DIDs makes logs instantly more interpretable for human auditors. The cryptographic link via DIDs ensures the identifier is not just a mutable string. The logged VCs provide the exact authorization context for the action, making audits far more precise.

- **Cryptographic Non-Repudiation of Agent Actions via DID Signatures:** To achieve strong non-repudiation, critical agent actions or the data they produce must be digitally signed by the agent using the private key associated with its DID. This is particularly important for actions with financial, legal, or safety implications.
    - **Scenario (A2A Context):** OrderPlacementAgent (did:com:retail:a2a:orderbot:v1.0, ANSName a2a://OrderPlacement.RetailTransactions.MegaCorp.v1.0.live) submits a purchase order to SupplierFulfilmentAgent (did:com:supplierX:a2a:fulfill:v2.1, ANSName a2a://Fulfilment.SupplyChain.SupplierX.v2.1.prod), which was discovered via ANS query for "SupplyChain.OrderFulfilment.SupplierX".
    - **A2A Message with Signed Payload and DID Context:** The OrderPlacementAgent constructs an A2A message. The core business payload (the order details) is signed.

      // A2A Message (Conceptual JSON representation)

```
{

 "a2aHeader": {

  "messageId": "order-uuid-554433",

  "senderId": "did:com:retail:a2a:orderbot:v1.0", // DID used as A2A identifier

  "recipientId": "did:com:supplierX:a2a:fulfill:v2.1", // Target DID

  "protocolVersion": "A2A/1.0",

  "timestamp": "2025-10-02T16:30:00Z"
```

```
 },
 "iamExtension": { // Our IAM framework's extension
  "verifiablePresentation": [ /* Optional: JWT of a relevant VC, e.g.,
"AuthorizedBuyerCredential" */ ]
 },
 "payload": { // This is the part that is primarily signed
  "orderId": "PO-2025-10-778",
  "items": [ {"sku": "XYZ123", "quantity": 100}, {"sku": "ABC789", "quantity": 50} ],
  "shippingAddress": "123 Main St, Anytown",
  "totalAmount": 12500.75,
  "currency": "USD"
 },
 "payloadSignature": { // Signature specifically over the 'payload' object
  "keyId": "did:com:retail:a2a:orderbot:v1.0#key-transact", // Specific key for
transactions
  "algorithm": "EdDSA",
  "signatureValue": "..." // Digital signature of canonicalized JSON payload
 }
}
```

- **Verification and Logging by SupplierFulfilmentAgent:**

  1. The AEM at SupplierFulfilmentAgent's side first authenticates the sender via its DID and any presented VCs (as per Section 5.1).
  2. It then specifically verifies the payloadSignature using the public key did:com:retail:a2a:orderbot:v1.0#key-transact (obtained by resolving the sender's DID).
  3. SupplierFulfilmentAgent's log entry for receiving this order would include: its own DID/ANSName, the sender's DID/ANSName, the order ID, a hash of the received payload, and the payloadSignature object. This creates a verifiable record that OrderPlacementAgent indeed sent that specific order. The initial discovery via

ANS ensures the order is sent to a legitimate fulfilment agent. The DID-based signature on the payload provides strong non-repudiation for the order's content, traceable to a specific, verifiable agent identity. Traditional EDI or API calls often rely on weaker authentication or channel security alone.

4. For high-assurance workflows, it is advisable to implement end-to-end payload integrity validation by cryptographically signing both request and response objects at each stage of the agent interaction chain.This approach ensures that any tampering whether at transport, storage, or application layers can be immediately detected. Verifying payload integrity upon receipt enhances both non-repudiation and forensic trust in audit logs.

- **Verifiable Provenance Chains in MCP Tool Interactions:** When an LLM-based agent uses an MCP tool, understanding the full chain, from user prompt to LLM, to MCP tool call, to tool result, back to LLM, and then to the user, is vital for auditing and debugging.

    - **Scenario:** A user asks ResearchLLM-Agent (did:com:ai-lab:mcp:researcher:zeta:v3.1, ANSName mcp://Researcher.ScientificQuery.AILab.v3.1.experimental) a complex question requiring a database lookup via an MCP tool, SemanticSearchTool (did:com:datastore:mcp:tool:semsearch:v1.0, ANSName mcp://SemanticSearch.KnowledgeBase.DataCorp.v1.0.main). ResearchLLM-Agent discovers SemanticSearchTool via an ANS query specifying the "KnowledgeBase.SemanticSearch" capability.

    - **MCP Interaction Logging with DIDs and VCs:**

        1. **User Interaction Log:** User prompt, timestamp, and ResearchLLM-Agent's DID/ANSName.
        2. **ResearchLLM-Agent Internal Log (or trace):**
            - Decision to use SemanticSearchTool.
            - Query sent to ANS for SemanticSearchTool.
            - Resolved DID/ANSName for SemanticSearchTool.
            - The MCP call it constructs to SemanticSearchTool, including:
                - Its own DID as the caller.
                - The JIT VC it obtained/presented for this tool use (e.g., vc:jwt:...:mcp-tool-access-zeta-job778).
                - The parameters sent to the tool.
            - This entire MCP call could be signed by ResearchLLM-Agent.
        3. **SemanticSearchTool (MCP Tool) Log:**
            - Its own DID/ANSName.
            - Receiving the MCP call from ResearchLLM-Agent (DID/ANSName logged).
            - The presented JIT VC ID.
            - Verification status of the caller's DID and VC.
            - Parameters received.
            - Actions it took (e.g., database queries it made internally).
            - The result it returned to ResearchLLM-Agent.

        - This log entry or the response payload could be signed by SemanticSearchTool.
    4. **ResearchLLM-Agent Internal Log (continued):**
        - Response received from SemanticSearchTool (potentially with signature verification).
        - How it processed the tool's output.
        - The final answer generated for the user (this answer could also be signed).

This chained logging, where each step is linked by verifiable DIDs/ANSNames and specific VCs or signed messages, creates a rich, end-to-end auditable provenance trail. If the final answer is wrong, auditors can trace back: Was it the LLM's reasoning leveraging timestamp of the LLM API calls, the MCP tool's execution, the data the tool accessed, or the initial ANS discovery that pointed to an incorrect tool version? This detailed, verifiable chain is crucial for explainability and accountability in complex agentic workflows involving external tools.

To enhance visibility and trust across agent interactions, enterprises should implement real-time behavioral monitoring within agent runtimes and interface layers. This includes detecting anomalies in output structure, execution timing, or response consistency, compared to historical baselines or allowed behavior policies.
Even when an agent presents valid cryptographic credentials, unexpected changes in behavior such as unusual API call sequences, elevated response latency, or semantic drift can indicate misuse or compromise.

Integrating these detection mechanisms into existing SIEM, agent telemetry streams, or dedicated runtime monitors enables early detection and containment of unauthorized behaviors, particularly in multi-agent or AI-powered workflows

- **Privacy-Preserving Audits of IAM Policies with ZKPs:** Organizations may need to prove to external auditors or regulators that their Agentic AI IAM policies are being correctly enforced, without revealing the proprietary details of all policies or all agent interactions.
    - **Scenario:** An auditor wants to verify that access to resources tagged PII_Strict is only ever granted if an agent presents a valid VC of type PII_AccessLevel3_Certified *and* the request originates from an approved network segment.
    - **Mechanism:**
        1. The IAM system's Policy Decision Point (PDP) logs all its decisions, including the agent DID, resource, action, presented VCs (or their hashes), contextual attributes, and the allow/deny outcome. These logs themselves could be cryptographically committed to (e.g., a hash chain).
        2. The organization can run a process that analyzes these logs and generates a ZKP. This ZKP would prove a statement like: "For all access requests to resources tagged PII_Strict within the last audit period that resulted in an 'allow' decision, the requesting agent's presented credentials included a valid (non-revoked, correctly signed) PII_AccessLevel3_Certified VC from an approved issuer, AND the source network attribute was in the set {'segA', 'segB'}."

3. This ZKP is generated *without revealing* the specific agent DIDs, resource DIDs, exact times, or other details of the individual access events.
4. The auditor receives and verifies this ZKP, along with information about the approved VC issuers and network segments, providing strong assurance of policy enforcement without seeing the raw, potentially sensitive log data. This enables "compliance as code" verification with privacy. It allows organizations to demonstrate adherence to internal or external IAM rules without exposing the minutiae of every transaction, which is a common challenge in traditional audit processes that often require extensive (and risky) data sharing.

By deeply integrating verifiable Agent IDs (DIDs/VCs), secure discovery via ANS, and cryptographic techniques like digital signatures and ZKPs into the logging and auditing process, our framework aims to create a system where agent actions are not just recorded, but are verifiably attributable, contextualized, and, where necessary, proven compliant in a privacy-respecting manner. This robust auditability is fundamental to building and maintaining trust in complex and autonomous MAS.

To align with data protection and compliance mandates, secure logging strategies should also define data retention lifecycles, auto-expiry policies, and access-controlled redaction mechanisms for sensitive logs. Ensuring that logs are not only immutable but also ephemerally governed strengthens both audit compliance and operational privacy in long-running agent environments.

## 5.3. Real-time Monitoring and Anomaly Detection

Effective IAM extends beyond static policy enforcement to encompass continuous, real-time oversight of agent activities. The rich, verifiable Agent IDs (DIDs and VCs), coupled with the contextual information available through Agent Name Service (ANS) resolutions, provide the foundation for a far more sophisticated and proactive monitoring and anomaly detection capability than achievable with traditional, opaque identifiers. This allows security systems to not only identify *what* is happening but also understand if it aligns with an agent's *intended and attested* purpose and capabilities.

- **Establishing Rich Behavioral Baselines Anchored to Verifiable Identities (DIDs and ANSNames):** Modern monitoring can move beyond tracking simple metrics like CPU usage per IP address. The proposed framework allows for the creation of multifaceted behavioral baselines for each unique agent DID and its associated ANSName profiles:
    - **Discovered vs. Declared Scope of Behavior:** The agent's DID Document contains its scopeOfBehavior (e.g., "customer_support_query_resolution_for_product_X"). ANS registration might also include a primary capability (e.g., Support.ProductQuery.CustomerFacing.v1). Monitoring systems can compare the agent's actual interactions and data access patterns against this declared and discoverable purpose. The current SIEM tool does not support this yet. There are some startup Agentic AI security companies actively working on this enhancement as an Agentic AI security posture management tool. Significant deviations trigger alerts.

        - *Scenario:* SupportAgentAlpha (did:com:support:agent:alpha01, ANSName helpdesk://Support.ProductQuery.CustomerFacing.v1.Acme) normally accesses the product knowledge base and customer ticket system. If it suddenly starts making frequent ANS queries for agents with FinancialData.InternalAudit capabilities, or attempts to access database schemas related to payroll, this is a strong anomaly relative to its declared/discovered scope.

    - **Authorized Toolset and ANS-Discoverable Service Usage:** The agent's DID Document details its toolset (specific APIs, other agent DIDs/ANSNames it's authorized to interact with). Monitoring systems can track:

        - Actual tool/API calls made.
        - ANS queries made by the agent to discover other services.
        - If the agent attempts to use tools not in its list or interact with DIDs/ANSNames that don't match its typical collaboration patterns or authorized interaction VCs.
        - *Scenario (MCP Context):* DataPipelineAgent-ETL (did:com:dataops:agent:etl04, ANSName mcp://ETL.DataWarehouseLoading.DataOps.v2. nightly) is authorized to use PostgresConnectorTool (an MCP tool discovered via ANS as mcp://DBConnector.PostgreSQL.InternalTools.v1.stable) and S3StorageTool. If it makes an ANS query for mcp://ExternalAPI.SocialMediaScraping... or attempts to invoke such a tool via MCP, it's a policy violation and an anomaly.

    - **VC Presentation Patterns:** Monitoring the types of VCs an agent typically presents for different actions, and the issuers of those VCs. An agent suddenly presenting a VC from a previously unseen or untrusted issuer for a high-privilege operation is suspicious.

    - **Communication Graph and Trust Dynamics:** Building a graph of typical agent-to-agent interactions (DID-to-DID or ANSName-to-ANSName) based on historical communication logs. New, unexpected communication links, especially with agents outside the organization or with low reputation scores (if a reputation system is integrated), can be flagged.

        - *Scenario:* A fleet of InventoryCheckAgent instances (e.g., a2a://InventoryCheck.RetailStoreXYZ.Ops.v1.hourly::did:...) typically only communicate via A2A with a central InventoryMasterAgent (a2a://InventoryMaster.HeadOffice.Ops.v3.main::did:...). If one InventoryCheckAgent initiates an A2A connection to an unknown external ANSName/DID, or starts sending unusually large A2A payloads, this is anomalous.

- **Advanced Deviation Detection Leveraging Verifiable Claims:** The ability to verify claims presented as VCs in real-time enhances anomaly detection:

    - **Scope Creep Beyond VC-Attested Capabilities:** An agent, ResearchSummarizer (did:..., ANSName a2a://Summarization.ScientificLiterature.ResearchGroup.v1), might hold a VC for "Access_PubMed_API_SummarizationOnly." If it attempts to use the PubMed API's "BulkDownloadAbstracts" function (which its VC does not authorize), the

AEM/PDP would block it, and the monitoring system would log this as a significant deviation, as it's attempting an action beyond its attested capability.

- **Anomalous JIT VC Requests:** If an agent frequently requests JIT VCs for tasks outside its typical operational parameters, or if the requested scopes for JIT VCs escalate without justification, this could indicate a compromised agent or a misbehaving workflow.
- **Interaction with Agents Lacking Expected Counter-Attestations:** If SecureDataTransferAgent is only supposed to send data to other agents that can present a "DataRecipient_EncryptionLevel5_Compliant" VC, an attempt to send data to an agent (discovered via ANS) that *cannot* present such a VC would be a flagged anomaly, even if basic network connectivity is possible.

- **Dynamic Trust Scoring and Risk-Adaptive IAM Incorporating ANS Context:** The Agent ID (DID) becomes the anchor for a dynamic trust score, influenced by monitoring. ANS context adds another layer.
  - **Inputs to Trust Score (with ANS context):**
    - Successful completion of tasks within the agent's ANS-declared capability.
    - Policy violations or anomalous behaviors (as detailed above).
    - Validity and issuer trustworthiness of its VCs.
    - Feedback from other reputable agent DIDs (whose own ANS profiles might indicate their roles/trustworthiness).
    - **ANS-related anomalies:** Repeatedly querying ANS for unrelated capabilities, attempting to register with a misleading ANSName, or interacting with agents resolved from suspicious ANS domains.
  - **Risk-Adaptive Policy Enforcement Example (A2A):** PaymentAgent-Acquirer (a2a://PaymentProcessing.Acquisition.FinServ.v2.live::did:...) normally processes transactions. It starts making unusual ANS queries for a2a://DataAggregation.UserProfiling... services and receives a few low-severity alerts for attempting to access non-payment related internal APIs.
    1. Its trust score, managed by the IAM system, is lowered.
    2. The Session Authority (SA) is notified of the trust score change.
    3. The SA updates the Session State Synchronizer (SSS) for this agent's global session, adding a "ReducedTrust" status or dynamically adjusting its permissible capability set.
    4. When PaymentAgent-Acquirer next attempts a high-value A2A payment authorization request to PaymentGateway-PSP (a2a://Gateway.PaymentAuth.PSPGlobal.v4.secure::did:...), the AEM at the gateway side consults the SSS.
    5. Even if the agent presents its usual VCs, the SSS indicates "ReducedTrust." The PDP at the gateway might now enforce a stricter policy: "IF agent_status == 'ReducedTrust', THEN require_multi_factor_agent_auth (e.g., a ZKP of a recent controller approval for this transaction type) OR limit_transaction_value_to_low_threshold." The A2A transaction might be rejected or queued for additional checks, preventing potential fraud by a slightly misbehaving or partially compromised agent.

The ANS provides discoverable context about an agent's *intended* role and capabilities. Monitoring deviations from this publicly or organizationally declared purpose, in addition to private policy violations, gives a richer signal for anomaly detection. The trust score becomes more robust as it can factor in the consistency of an agent's behavior with its registered identity profile in ANS.

## 5.4. Agile Incident Response: Precision Targeting, Rapid Containment, and Discoverable Impact

When a security incident occurs, the ability to respond swiftly, precisely, and comprehensively is critical to minimizing damage. The proposed IAM framework, with its integration of DIDs, VCs, and ANS, provides superior capabilities for incident response.

- **Rapid and Unambiguous Identification via DID and ANS Context:** Security alerts from monitoring systems or external threat intelligence will directly reference the compromised or malicious agent's DID and often its ANSName. This removes ambiguity and allows response teams to immediately identify:
    - The specific agent instance involved (via DID).
    - Its declared purpose and owner (via ANSName and resolved DID document).
    - Its attested capabilities and dependencies (via VCs and DID document).
    - *Example:* An alert "Unusual data exfiltration by did:com:cloudstorage:agent:backup-beta-721 (ANSName: a2a://Backup.CriticalDB.AcmeCorp.v1.beta. nightly)" immediately tells the SOC:
        - It's a specific backup agent instance.
        - It's associated with AcmeCorp's critical database backups.
        - It's a beta version (which might imply higher risk or different oversight).
- **Targeted Revocation with Ecosystem-Wide Propagation:** The framework supports granular to broad revocation, propagated efficiently:
    - **VC Revocation (Surgical):** If a specific attested capability (e.g., VC:AbilityToModifyUserPermissions) of AdminBot-HR (did:com:hr:adminbot:003, ANSName a2a://UserAdmin.Permissions.HRInternal.v2.prod) is found to be exploited due to a bug, that VC is added to a VC Status List. AdminBot-HR might still function for other tasks (e.g., reading user profiles) using its other VCs, but attempts to use the revoked permission VC will fail.
    - **DID Deactivation/Revocation (Logical via DID Method or ANS):** If AdminBot-HR's private keys are confirmed stolen, its entire DID (did:com:hr:adminbot:003) is revoked via its DID method. The ANS entry for a2a://UserAdmin.Permissions.HRInternal.v2.prod would then either resolve to a "revoked" status or be removed/updated by the ANS Registration Authority. Other agents querying ANS for this service will no longer receive the compromised DID.

- **Instantaneous Global Session Invalidation via Unified Enforcement Layer:** This is the most critical response.

    1. **Trigger:** SOC confirms did:com:hr:adminbot:003 is actively malicious.
    2. **SA Notification:** The Session Authority (SA) is notified, specifying the DID.
    3. **SSS Update:** SA updates the Session State Synchronizer (SSS) to mark all global sessions for did:com:hr:adminbot:003 as "TERMINATED_IMMEDIATE_SECURITY_LOCKOUT".
    4. **AEM Enforcement:**
        - All AEMs interacting with or receiving requests from did:com:hr:adminbot:003 (whether via A2A, MCP, or internal ACP/HTTP calls) consult the SSS.
        - They see the "TERMINATED" status and instantly block any new requests and terminate any active local protocol sessions.
        - *Scenario (MCP Tool in use by AdminBot-HR):* If AdminBot-HR was using an MCP tool like UserProvisioningTool, its active MCP session (managed by the tool's AEM) would be killed. Further MCP calls from AdminBot-HR would be rejected by the AEM before even reaching the tool's logic.
        - *Scenario (A2A communication):* If AdminBot-HR was sending A2A messages to AuditLogAgent, these A2A messages would be blocked by the AEM on AuditLogAgent's side.

    The ANS provides a clear point for signaling revocation at the discovery layer. Even if an attacker has cached an old DID, new discovery attempts for the agent's function would fail or return a revoked status. The SSS ensures that active sessions, regardless of how they were initiated (perhaps post-ANS discovery), are comprehensively terminated.

- **Rich Forensic Analysis with Discoverable Context:** Post-incident, the combination of DID-anchored logs, VCs, and ANS information provides unparalleled depth for forensics.

    - **Contextualizing Compromise:** If did:com:research:agent:dataminer:gamma-9 is compromised, investigators can not only see its actions (via DID logs) but also:
        - Resolve its ANSName (science://DataMining.LargeDatasets.ResearchDiv.v0.9.experimental) to understand its expected role and provider context.
        - Examine its DID Document and VCs to see its intended capabilities and dependencies (e.g., "depends on did:com:lib:math:vectorcalc:v3.2"). This helps to check if a *dependency* was the root cause.
        - Trace its ANS query history: Was it trying to discover and interact with services outside its normal profile before the compromise?
        - If it interacted with other agents, their DIDs/ANSNames are in the logs, allowing investigators to assess the blast radius and check if those collaborators were also affected or were part of the attack.

    - **Identifying Attack Vectors via ANS:** If multiple agents registered under a specific, less reputable Provider in their ANSNames are simultaneously compromised, it might indicate

a targeted attack against that provider's agent infrastructure or a vulnerability common to their agents.

ANS data (like provider, capability domain in the name) adds valuable metadata for clustering incidents, identifying patterns, and understanding the potential scope or origin of an attack that might involve multiple agent instances from a similar source or with similar functions.

## 5.5. Other Potential Uses Building on Verifiable Agent IDs and Discoverable ANS Profiles

The synergistic use of detailed, verifiable Agent IDs and a structured Agent Name Service, all managed within a robust IAM framework, naturally extends to enable further advanced functionalities critical for a mature and trustworthy AI ecosystem.

- **Decentralized Reputation and Trust Brokering with ANS-Contextualized Feedback:**
    - Agent DIDs serve as the stable anchors for accumulating reputation scores. When AgentA (e.g., discovered via ANS as a2a://TaskExecutor.GeneralPurpose.CommunityPool.v1.standard::did:agentA...) completes a task for AgentB, AgentB can issue a reputation VC attesting to AgentA's performance, timeliness, and reliability for that specific task type (derived from AgentA's ANS capability).
    - These VCs can be stored by AgentA or published to a decentralized reputation ledger. Future agents querying ANS for "TaskExecutor.GeneralPurpose" might then also be able to query this reputation system (using the resolved DID) for community feedback, prioritizing agents with higher, relevant reputation scores. The ANS capability string itself provides context for the reputation (e.g., good at "GeneralPurpose" tasks).
    - *Code Concept: Agent B issuing a reputation VC for Agent A:*

```
# Agent B's perspective

from pyld import jsonld # For Verifiable Credentials

from did_sdk import sign_vc # Conceptual SDK function

agent_A_did = "did:agentA..."

agent_A_ans_capability = "TaskExecutor.GeneralPurpose.CommunityPool.v1.standard"

reputation_claim = {

  "@context": ["https://www.w3.org/2018/credentials/v1",
"https://example.org/reputation/v1"],
```

```
    "type": ["VerifiableCredential", "ReputationCredential", "PerformanceReview"],
    "issuer": "did:agentB...", # Agent B's DID
    "issuanceDate": "2025-10-03T10:00:00Z",
    "credentialSubject": {
        "id": agent_A_did,
        "ansCapabilityContext": agent_A_ans_capability,
        "rating": 5, // Scale of 1-5
        "comment": "Completed task efficiently and accurately.",
        "taskId": "task-uuid-for-context"
    }
}
# Agent B signs this claim with its DID key to create a VC
signed_reputation_vc = sign_vc(reputation_claim, "did:agentB...", "did:agentB...#key-1")
# Agent B might then send this VC to Agent A, or publish it to a reputation service.
```

- **Automated Billing and Resource Quota Enforcement via ANS-Defined Services:**
    - When an agent discovers and uses a commercial service (e.g., a specialized MCP tool like mcp://AdvancedTranslation.Multilingual.PremiumAPI.v3.commercial::did:tool:translateXYZ...) via ANS, the ANS record itself might point to metadata about pricing models or rate limits associated with that service DID.
    - The consuming agent's DID is logged by the commercial tool for every API call. The tool provider's AEM/PDP can enforce quotas (e.g., "Agent did:com:startup:agent:translator007 has a quota of 10M characters/month for did:tool:translateXYZ"). Billing is then accurately attributed to the consuming agent's controller.

- **Secure Software/Model Supply Chain Attestations Linked to ANS Registrations:**
    - When an agent is registered with ANS (e.g., a2a://ImageRecognition.MedicalScans.RadAI.v2.validated::did:radai:imgrec:002), part of its registration with the ANS Registration Authority (RA) could involve presenting VCs that attest to its supply chain security:

        - A VC for its base foundation model (e.g., "ModelCard_VC_for_RadAI_BaseVisionModel_v2"), detailing its training data, bias tests, and safety evaluations.
        - SBOM VCs for its software components.
        - A "ValidatedSecureBuild_VC" from a trusted CI/CD pipeline.
    - The ANS resolver could then optionally return indicators of these attestations (or links to the VCs) along with the agent's DID, allowing discoverers to prioritize agents with verifiable supply chain security.

- **Dynamic Coalition Formation and Capability Negotiation using ANS for Initial Matching:**
    - An EmergencyResponseOrchestratorAgent queries ANS for agents with diverse capabilities like a2a://DroneSurveillance.DisasterZoneMapping..., mcp://Logistics.ResourceAllocation..., and comms://TemporaryNetwork.MeshDeployment....
    - Once candidate DIDs are retrieved, the orchestrator can initiate a negotiation phase (e.g., using FIPA Contract Net Protocol messages over A2A or ACP). During negotiation, agents exchange more detailed VCs about their specific sub-capabilities, current availability, and resource needs.
    - The orchestrator then issues a "CoalitionCharter_VC" to the selected agents, defining the coalition's DID, its mission, shared resources (perhaps managed by a temporary group DID), roles, and duration. This VC acts as a temporary authorization within the coalition.

- **ANS for Discovering Ethical AI Governance Services:**
    - Agents or users could query ANS for services like audit://EthicalComplianceOracle.AIBehavior.IndependentOrg.v1 or report://BiasReportingService.FairnessConsortium.v1.
    - These specialized services (themselves having DIDs and VCs) could then be used by agents to self-assess their decisions against ethical guidelines or for users to report problematic agent behavior, with the AN DIDs providing a verifiable link to the service.

By integrating ANS as a core discovery mechanism whose results (DIDs, initial capability claims) feed directly into the DID/VC-based authentication and authorization processes, the entire IAM lifecycle becomes more context-aware, secure, and efficient. The discoverable nature of agent capabilities and attestations fosters a more transparent and trustworthy ecosystem.

# 6. Deployment Models & Governance Considerations

The proposed Agentic AI IAM framework, while architecturally comprehensive, is not a monolithic, one-size-fits-all solution in terms of its practical implementation. The diverse needs of different organizations, Multi-Agent System (MAS) scopes (private enterprise vs. open ecosystem), trust

requirements, and existing infrastructure will necessitate different deployment models for its core components (e.g., DID registries, Verifiable Credential (VC) issuers, Agent Name Service (ANS), Policy Engines, Session Authority, Session State Synchronizer). Furthermore, regardless of the chosen deployment model, robust, well-defined, and adaptable governance is paramount for the long-term viability, trustworthiness, security, and interoperability of any such advanced IAM system. Governance frameworks should define/include policies for ephemeral identity lifecycle management, including instantiation, purpose declaration, and deactivation.

## 6.1. Deployment Model Analysis

We analyze three primary deployment models,Centralized, Decentralized, and Federated,assessing their characteristics, advantages, disadvantages, and suitability for various Agentic AI IAM scenarios.

### 6.1.1. Centralized Approach

- **Description:** In a centralized deployment, a single organization, platform provider, or a designated administrative entity controls and operates all, or the significant majority, of the IAM framework's core components. This typically includes:
    - The primary Agent ID registry (which might be a private Public Key Infrastructure (PKI) issuing X.509 certificates as per some ANS proposals, a proprietary database issuing unique identifiers, or a private DID method controlled by the organization).
    - The authoritative VC issuers for organizational roles, capabilities, and compliance attestations.
    - The ANS, if implemented as a private or enterprise-scoped directory service.
    - The central Policy Decision Points (PDPs) and Policy Administration Points (PAPs) define and enforce access rules.
    - The Cross-Protocol Session Authority (SA) and the Session State Synchronizer (SSS). Agents operating within this model typically belong to, or are tightly managed and permissioned by, the central entity. All trust decisions ultimately flow from this central authority.
- **Advantages:**
    - **Simplified Governance & Policy Cohesion:** Policy definition, updates, enforcement rules, and dispute resolution are managed by a single authority, leading to consistent application and rapid changes if needed.
    - **Unified Control, Visibility, and Audit:** Easier to monitor all agent activity, audit compliance comprehensively, and implement system-wide security updates or revocations efficiently.
    - **Potentially Easier Integration with Existing Enterprise Systems:** Can be more straightforward to integrate with existing enterprise IAM (e.g., Azure AD, Okta for human controllers/admins), logging infrastructure, and internal PKI.

    - **Optimized Performance:** Centralized components can often be tuned and optimized for performance within a known, controlled network environment.
    - **Clear Accountability:** Lines of responsibility for IAM operations, security incidents, and data stewardship are generally unambiguous.

- **Disadvantages:**

    - **Single Point of Failure, Control, and Trust:** The central entity becomes a critical dependency. Its compromise, outage, or policy failure can cripple the entire MAS IAM.
    - **Scalability Bottlenecks:** While optimizable, a purely centralized system can face significant scalability challenges as the number of agents, interactions, and policy evaluations grows into the millions or billions.
    - **Vendor/Platform Lock-in:** If the centralized IAM is tied to a specific vendor's proprietary implementation, switching platforms or interoperating with external systems that use different IAM models can be difficult and costly.
    - **Limited Cross-Organizational Trust & Interoperability:** Inherently less suitable for scenarios where agents from different, mutually untrusting organizations need to collaborate directly as peers. It requires all external parties to place their trust in, and often conform to the policies of, the central operator.
    - **Potential for Censorship or Abuse of Power:** The central authority has significant power over agent identities and access, which could be misused.

- **When to Use:**

    - **Enterprise-Internal MAS:** For AI agents owned and operated entirely within a single organization for internal automation, private AI-powered services, or employee-facing tools.
    - **Specific AI Platforms:** Provided by a vendor that offers a managed, walled-garden environment for their agents, handling IAM as an integrated part of the platform offering (e.g., a cloud provider's agent-building service).
    - **Early-Stage Deployments or Controlled Experiments:** Where simplicity of management, rapid iteration, and direct control are prioritized over decentralized trust or broad interoperability.
    - **Highly Regulated Environments with a Single Auditing Authority:** Where a central point of control and audit is mandated.

### 6.1.2. Decentralized Approach

- **Description:** Core IAM components are implemented using decentralized technologies, often public and permissionless, or permissioned consortia-based Distributed Ledger Technologies (DLTs). Key characteristics include:

    - DIDs are registered on public or consortia DLTs (e.g., did:ion, did:ethr, did:sov, or a custom agent-focused DID method on a dedicated ledger like the proposed Agent ID Provider Network - AIPN). Agent controllers or agents themselves manage their DID's private keys.

    - VCs can be issued by a diverse set of issuers (each with their own DID) and their status (revocation) might be tracked via decentralized mechanisms (e.g., on-chain registries, distributed VC status lists).
    - ZKPs are used extensively for privacy-preserving presentation of VCs and attributes.
    - ANS could be built on decentralized name systems (e.g., ENS, Handshake, or a custom DLT-based ANS).
    - Policy enforcement might involve smart contracts acting as rudimentary PDPs for on-chain resources, or rely on Verifiable Presentations that bundle VCs required by a verifier's policy. Global session state (like revocation lists) might be mirrored on resilient DLTs.
    - Governance is typically community-driven (e.g., DAOs for protocol upgrades) or based on the immutable logic encoded in smart contracts.

- **Advantages:**

    - **No Single Point of Failure or Control:** Enhanced system resilience; no single entity can unilaterally take down the identity system or censor participants.
    - **User/Agent Sovereignty (SSI):** Aligns strongly with Self-Sovereign Identity principles, giving agent controllers maximum control over their agents' identities, data, and disclosures.
    - **Enhanced Trust in Open, Permissionless Ecosystems:** Can foster greater trust in interactions between previously unknown parties, as identity claims are anchored on immutable, publicly verifiable ledgers (for public DLTs).
    - **Transparency & Auditability (for public DLTs):** DID registrations, VC schema registrations, and potentially high-level policy commitments can be publicly auditable.
    - **Censorship Resistance:** More difficult for any single entity to deplatform agents or deny them identity services.

- **Disadvantages:**

    - **Governance Complexity & "Tragedy of the Commons":** Achieving consensus on standards, operational policies, issuer accreditation, dispute resolution, and funding in a fully decentralized manner is extremely challenging.
    - **Smart Contract and DLT Security Risks:** Vulnerabilities in the underlying DLT protocol or the smart contracts implementing DID methods, VC registries, or policy logic can have widespread and often irreversible consequences.
    - **Performance, Scalability, and Cost of DLTs:** Many DLTs (especially public permissionless ones) face inherent limitations in transaction throughput, latency, and cost per transaction, which could be prohibitive for high-frequency IAM operations (e.g., JIT VC issuance, rapid session updates at scale).
    - **User/Controller Experience (Key Management):** Securely managing private keys for DIDs in a decentralized setting, without relying on a central custodian, can be a significant burden and risk for users or organizations controlling agents. Meanwhile, This can also be an advantage. The ability to manage your own keys in a separate environment from a central custodian can reduce attack surface.
    - **Irreversibility and Data Privacy:** Data written to immutable ledgers is extremely difficult (or impossible) to remove, posing challenges for "right to be forgotten"

requirements under regulations like GDPR, or for correcting erroneous identity information. Careful design of what goes on-chain versus off-chain is critical.
    - **Bootstrapping Trust:** Establishing initial trust in a new decentralized network of issuers and verifiers can be difficult without recognized authorities or a critical mass of reputable participants.

- **When to Use:**
    - **Truly Open, Permissionless Multi-Agent Ecosystems:** Where agents from any origin can participate and interact as peers (e.g., decentralized social media, open marketplaces for AI services, global scientific collaboration platforms).
    - **Cross-Organizational Collaborations Without a Central Trusted Party:** When participating organizations are peers and no single entity can or should act as the central IAM authority.
    - **Applications Requiring Very High Degrees of Censorship Resistance or User Control Over Identity:** Such as tools for activism, journalism in repressive environments, or personal data stores controlled by individual AI agents.
    - **Ecosystems Where a Transparent, Community-Governed Trust Infrastructure is a Core Design Goal.**

### 6.1.3. Federated Approach

- **Description:** This model involves multiple independent IAM domains or "trust communities." Each domain might manage its own IAM infrastructure using centralized or even localized decentralized approaches. The key is that these domains establish mutual trust relationships and define standardized protocols for interoperability. This could involve:
    - Cross-certification of Certificate Authorities (CAs) or DID method roots between domains.
    - Shared trust lists for recognized VC issuers and verifier policies across the federation.
    - Federated ANS resolution (e.g., similar to how DNS subdomains can be delegated, or using inter-registry lookup protocols).
    - Use of highly interoperable DID methods and standardized VC profiles (e.g., based on W3C specs) to ensure credentials from one domain can be understood and verified in another.
    - A central (or mutually agreed upon) body might define the "federation rules" or baseline interoperability standards, but day-to-day IAM within each domain remains autonomous.

- **Advantages:**
    - **Balances Autonomy with Interoperability:** Organizations or communities can maintain control and sovereignty over their own IAM policies, infrastructure, and agent populations while still enabling their agents to securely interact with agents from other trusted domains in the federation.

    - **Scalability:** Scales effectively by distributing the IAM load and management responsibilities across multiple autonomous domains. Avoids the bottlenecks of a single global centralized system.
    - **Domain-Specific Policies and Trust Levels:** Allows IAM policies and trust requirements to be tailored to the specific needs, risk appetite, and regulatory context of different industries, communities, or legal jurisdictions within the federation.
    - **Enhanced Resilience:** Failure or compromise within one federated domain does not necessarily bring down the entire system or affect the security of other independent domains (assuming proper isolation and trust boundary enforcement).
    - **Phased Adoption & Existing System Integration:** Can allow organizations with established IAM to join a federation by implementing "bridge" services or adapters, rather than requiring a full rip-and-replace.

- **Disadvantages:**

    - **Complexity of Trust Management:** Establishing, maintaining, updating, and revoking trust relationships between multiple autonomous domains is technically and politically complex. Managing shared trust roots, evolving policy mappings, and ensuring consistent liability frameworks requires significant effort.
    - **Interoperability Challenges (Technical and Semantic):** Ensuring that identity information, VC schemas, policy languages, and revocation signals are truly interoperable across different domains (which might use different underlying technologies) requires rigorous adherence to common standards and ongoing coordination. Semantic mismatches in attribute definitions can lead to misinterpretations.
    - **Potential for Lowest Common Denominator Security:** If the criteria for joining the federation or for mutual trust acceptance between domains are too lax, it can inadvertently weaken the overall security posture of all participants who trust that domain.
    - **Discovery and Pathfinding Complexity:** Discovering agents or resolving identity information across multiple federated domains can be more involved than within a single, unified domain, potentially requiring multi-hop lookups or reliance on a federated discovery service.
    - **Governance Overhead for the Federation Itself:** A body or process is needed to govern the federation's rules, membership, standards, and dispute resolution mechanisms between domains.

- **When to Use:**

    - **Consortia of Organizations in a Specific Industry:** E.g., financial institutions forming a network for secure inter-agent transactions, healthcare providers for federated health data exchange via agents, supply chain partners for collaborative logistics.
    - **Alliances of Research Institutions or Governmental Agencies:** Sharing data or computational resources via AI agents across organizational boundaries, according to agreed-upon rules.
    - **Large, Multi-National Corporations with Distinct Regional or Business Unit IAM Requirements:** Where each unit manages its local agent IAM but needs secure global inter-unit agent interaction.

- **Ecosystems Evolving from Existing Centralized or Siloed Systems Towards Greater Interoperability:** Where a full move to decentralization is not feasible or desired, but interoperability is key.
- **As a practical model for the Agent Name Service (ANS):** Different organizations could run their own ANS "zones" for their agents but participate in a federated resolution system.

### 6.1.4. Hybrid Approaches:

It's important to note that these models are not always mutually exclusive. Hybrid approaches are likely to be common, combining elements from each.

- *Example 1:* An enterprise might use a centralized IAM framework for its internal agents but use a federated model to interact with agents from trusted partners. Its internal agents might have DIDs issued by a private DID method, but these DIDs could be anchored or discoverable through a broader federated system.
- *Example 2:* A decentralized ecosystem might still rely on a few, highly reputable (perhaps foundation-run) "anchor" VC issuers for certain critical credentials (like "VerifiedLegalEntity_VC"), even if most other VCs are issued more peer-to-peer.
- *Example 3:* The Session Authority and Session State Synchronizer, while logically providing global coordination, might be implemented as a permissioned DLT operated by a consortium (federated control over a logically centralized function) for resilience and shared trust.

## 6.2. Decision Matrix for Choosing an Implementation Model

Selecting the most appropriate deployment model requires careful consideration of various factors. The following matrix provides guidance:

| Feature / Requirement | Centralized | Decentralized | Federated | Hybrid |
|---|---|---|---|---|
| **Control & Authority** | Single Entity (High Control) | Community/Protocol (Low Central Control) | Domain-Specific + Federation Body (Balanced) | Varies; often domain-specific with shared elements |
| **Trust Model** | Hierarchical (Trust in Central Entity) | Peer-to-Peer / Ledger-Based (Distributed Trust) | Inter-Domain Agreements / Shared Roots (Delegated) | Mix of hierarchical and delegated/distributed |

| Feature / Requirement | Centralized | Decentralized | Federated | Hybrid |
|---|---|---|---|---|
| **Scalability** | Moderate (Potential Bottlenecks) | Potentially Very High (if DLT scales) / Variable | High (Distributed across domains) | High (Can optimize components) |
| **Performance (Latency)** | Potentially Low (if optimized, local) | Variable (DLT dependent, often higher) | Moderate (Inter-domain hops) | Variable (Can optimize critical paths) |
| **Interoperability (External)** | Low (Proprietary by default) | Potentially High (if open standards used) | High (Designed for inter-domain ops) | Moderate to High (Depends on bridge design) |
| **Complexity of Setup** | Low to Moderate | High | High (Trust agreements complex) | Moderate to High |
| **Complexity of Governance** | Low (Single decision-maker) | Very High (Consensus, community) | High (Federation rules, inter-domain) | High (Managing diverse components) |
| **Cost (Infrastructure)** | Moderate (Centralized infra) | Variable (DLT fees can be high) | Moderate to High (Per-domain + federation infra) | Variable |
| **Security (vs External Threats)** | Single attack surface (high impact if breached) | Distributed risk, smart contract vulns critical | Risk shared/isolated per domain; inter-domain trust | Tailorable; can have strong internal, defined external |
| **User/Agent Sovereignty** | Low | Very High | Moderate (Within domain policies) | Variable |
| **Censorship Resistance** | Low | High | Moderate (Per domain) | Variable |
| **Privacy Preservation** | Dependent on central entity's policies | High (with ZKPs, careful DLT use) | Domain-specific policies; inter-domain data flow | Can be designed for high privacy |
| **Suitability: Enterprise Internal** | **High** | Low | Moderate (For large, distinct internal units) | **High** (Central core, federated edges) |

| Feature / Requirement | Centralized | Decentralized | Federated | Hybrid |
|---|---|---|---|---|
| **Suitability: Open Ecosystem** | Low | **High** | Moderate (Federation of open communities) | Moderate (Public services with private backends) |
| **Suitability: B2B Consortia** | Low (Unless one org dominates) | Moderate (If common DLT agreed) | **High** | **High** (Federated interfaces, shared services) |

*Table 4: Deployment Model Comparison Matrix for Multi-Agent IAM Systems*

**How to use the matrix:**

1. Identify the primary context for your MAS (e.g., internal enterprise, open research platform, industry consortium).
2. Prioritize your key requirements (e.g., is maximum agent sovereignty critical, or is centralized auditability paramount?).
3. Evaluate each model against your high-priority requirements.
4. Consider if a hybrid approach offers the best trade-offs by combining strengths of different models for different IAM components (e.g., decentralized DIDs but a federated or even centrally managed Session Authority for specific use cases).

## 6.3. Governance Considerations:

Effective governance is the bedrock upon which trust and interoperability in any Agentic AI IAM framework are built. It's not merely about technical rules but also about establishing clear roles, responsibilities, processes for decision-making, dispute resolution, and adaptation over time.

- **Identity Governance (DIDs, VCs, ANS):**
    - **DID Method Governance:** For any DID method used (especially custom or DLT-based ones), there must be a clear governance framework detailing how the method is maintained, upgraded, how its security is overseen, and how operational parameters (like fees, if any) are set. For public DID methods, this is often managed by the respective communities or foundations.
    - **ANS Namespace Management & Policy:**
        - **Registration Authority (RA) for ANS:** Defining the roles and responsibilities of RAs that approve ANSName registrations. This includes criteria for validating the requester's legitimacy to claim a particular Provider or agentCapability namespace within ANS.

        - **Dispute Resolution:** Establishing impartial mechanisms for resolving conflicts over ANSNames (e.g., two entities claiming the same Provider.agentCapability combination). This might involve arbitration panels or community voting, depending on the governance model.
        - **Reserved Namespaces:** Potentially reserving certain top-level agentCapability domains or Protocol identifiers to prevent conflicts and ensure semantic clarity, managed by a standards body or a governance council.

    - **VC Issuer Accreditation, Trust Registries, and Governance Frameworks:**

        - **Issuer Qualification:** Defining criteria for entities to become trusted VC issuers for specific types of claims (e.g., what qualifications does an entity need to issue a "SOXComplianceCertified" VC vs. a "CommunityReputation_Level5" VC?).
        - **Trust Registries:** Maintaining discoverable registries of accredited VC issuers, their DIDs, the types of VCs they are authorized to issue, and potentially their own compliance/audit status. These registries themselves must be governed securely.

        - **VC Schema Governance:** Processes for proposing, standardizing, versioning, and decommissioning VC schemas to ensure semantic interoperability.

    - **Agent ID Lifecycle Management Policies:** Documented policies detailing the processes for agent ID registration (including identity proofing of the controller), regular renewal/re-attestation requirements, conditions for suspension (e.g., due to suspicious activity) vs. full revocation (e.g., confirmed compromise), and data remanence considerations upon decommissioning an agent ID.

- **Security Policy Governance (for PDPs and SA):**

    - **Policy Authorship & Approval Workflows:** Clear processes for who can define, review, test, and approve access control policies that are loaded into PDPs or enforced by the Session Authority. This should involve multiple stakeholders, including security teams, business owners, and legal/compliance.
    - **Policy-as-Code Principles:** Managing policies using version control systems, automated testing (e.g., unit tests for policy logic, integration tests against simulated agent requests), and CI/CD pipelines for deployment to ensure consistency and auditability.
    - **Emergency Policy Override Procedures:** Well-defined and highly restricted procedures for emergency overrides of policies in critical situations, with mandatory post-incident review and justification.
    - **Policy Interoperability/Harmonization (in Federated Models):** Establishing baseline policy requirements or mapping frameworks to ensure a minimum level of consistent security across federated domains.

- **Operational and Security Governance for IAM Infrastructure:**

    - **Incident Response Playbooks for IAM Breaches:** Specific playbooks for responding to compromises of core IAM components (e.g., a compromised VC issuer DID, a DoS

attack on the SSS, a vulnerability in the PDP software). This includes communication plans for affected parties.

- **Key Management Governance for IAM Services:** Strict policies and procedures for the generation, storage (e.g., mandatory HSM use for SA signing keys), rotation, and destruction of cryptographic keys used by the IAM services themselves.
- **Regular Audits & Penetration Testing:** Mandating periodic independent security audits and penetration tests of the core IAM infrastructure components and protocols.
- **Vulnerability Disclosure Policy:** A clear policy for how vulnerabilities discovered in the IAM framework or its components should be reported, triaged, and remediated.
- **Cross-Environment Governance:** Cross-environment security policies should include mechanisms for cross-tenant isolation, federated signature validation, and audit harmonization across multi-cloud or hybrid deployments. These steps ensure policy enforcement remains consistent, traceable, and fault-tolerant, even when agent activities span infrastructure or trust domains.

- **Data Privacy and Ethical Use Governance:**

  - **Data Protection Impact Assessments (DPIAs) for IAM Data:** Conducting DPIAs for the IAM framework itself, considering the personal data (e.g., controller DIDs, VCs linking agents to potentially sensitive tasks) it processes and stores.
  - **Agent ID Data Minimization Principles:** Guiding agent developers and controllers to only associate the minimum necessary attributes and VCs with an Agent ID for its intended purpose.
  - **Bias Review in Credentialing and Reputation:** Establishing processes to review VC issuance criteria and reputation system algorithms for potential biases that could unfairly disadvantage certain agents or their controllers.
  - **Ethical Oversight Bodies:** Potentially establishing an ethics council or review board to oversee the evolution of the IAM framework, consider novel ethical challenges posed by agent identities, and provide guidance on responsible use.

- **Evolution and Standards Governance:**

  - **Change Management Process:** A formal process for proposing, debating, approving, and implementing changes or upgrades to the core IAM framework protocols, schemas, and governance rules. This should be transparent and allow for community/stakeholder input.
  - **Liaison with External Standards Bodies:** Maintaining active engagement with relevant standards bodies (W3C, IETF, DIF, etc.) to ensure the framework aligns with and contributes to broader digital identity and security standards.

Effective governance in the Agentic AI IAM space will not be static; it must be an adaptive system capable of evolving alongside the technology and the threat landscape. It necessitates a collaborative effort, potentially involving a mix of industry self-regulation, standards development, and, where appropriate, governmental oversight, particularly for public-facing or critical infrastructure components.

# 7. Security Considerations

Securing the Agentic AI IAM framework is paramount, analyzed here using the MAESTRO framework (Huang, 2025b). The rapid adoption of Agentic AI systems introduces a complex threat landscape with new attack vectors, as meticulously mapped by the MAESTRO Agentic Threat modelling framework. Our alignment of MAESTRO threats with MITRE D3FEND countermeasures reveals a critical insight: while AI presents novel attack vectors, effective AI security is fundamentally rooted in the rigorous and adaptive application of established cybersecurity principles. This security considerations section outlines key strategic considerations to guide organizations in building, deploying, and securing resilient Agentic AI systems by leveraging MAESTRO threat modelling framework.

## 7.1 The MAESTRO 7-Layer Reference Architecture for Agentic AI

MAESTRO decomposes AI ecosystems into: Layer 1: Foundation Models, Layer 2: Data Operations, Layer 3: Agent Frameworks, Layer 4: Deployment and Infrastructure, Layer 5: Evaluation and Observability, Layer 6: Security and Compliance (Vertical), and Layer 7: Agent Ecosystem.

## 7.2 Threat Analysis of the Proposed Agentic AI IAM Framework using MAESTRO Layers

- **L1: Foundation Models:** Model-based identity theft that occurs when attackers use AI models to analyze and replicate the behavioral patterns, communication styles, and decision-making characteristics of legitimate agents, effectively creating digital impersonators that can fool other systems or users into believing they're interacting with the authentic agent.(mitigated by cryptographic DIDs/VCs). In practice an attacker who clones an agent's model and steals its keys could still sign malicious actions. The framework should address how to distinguish an impostor that has valid keys. For example, are there dynamic proofs or challenge–response protocols to verify "liveness" of an agent. See our another paper for details on how to deal with this kind of attack: https://arxiv.org/abs/2506.13590
- **L2: Data Operations:** Poisoning of DID registries/VC status lists (mitigated by DLT consensus, signed registry entries); exfiltration of identity data (mitigated by encryption, agent-held VCs, ZKPs); tampering with PIPs (mitigated by PIP identity and secure channels).
- **L3: Agent Frameworks:** Compromised IAM SDKs (mitigated by secure development, sandboxing); framework vulnerabilities allowing session hijacking (mitigated by continuous re-validation via AEM/SSS).
- **L4: Deployment and Infrastructure:** DoS/DDoS against IAM services (mitigated by standard defenses, resilient design); compromise of IAM service infrastructure (mitigated by hardening,

access controls); lateral movement to IAM components (mitigated by network segmentation, Zero Trust).

- **L5: Evaluation and Observability:** Tampering with IAM audit logs (mitigated by immutable logging, signatures); evasion of IAM monitoring (mitigated by comprehensive instrumentation); data leakage via observability tools (mitigated by masking, ZKPs).
- **L6: Security and Compliance:** Misconfiguration of IAM policies (mitigated by policy-as-code, audits); compromise of IAM service keys (mitigated by HSMs, revocation); non-compliance with privacy regulations (mitigated by privacy-by-design, ZKPs).
- **L7: Agent Ecosystem:** Agent impersonation/DID spoofing (mitigated by cryptographic verification, VC status checks); compromised ANS leading to malicious discovery (mitigated by secure ANS resolution); collusion to falsify VCs (mitigated by trust diversification, reputation systems).

Threat Modeling and Mitigations based on MAESTRO framework

| No | Threat | Description | MEASTRO Layer(s) Impacted | Risk Impact | Recommended Mitigation |
|---|---|---|---|---|---|
| 1 | DID Spoofing/ Impersonation | Forged or manipulated DID Documents to impersonate privileged agents | **L3 (Agent Frameworks)** – Agent identity logic **L6 (Security & Compliance)** – Identity verification controls **L7 (Agent Ecosystem)** – Trust in agent discovery and reputation | Unauthorized access; data exfiltration | Resolver pinning, Validating signatures of DID Documents |
| 2 | VC Replay | Reuse of stolen Verifiable Credentials to perform privilege escalation | **L3 (Agent Frameworks)** – Credential issuance and validation **L6 (Security & Compliance)** – Token lifecycle governance | Privilege Escalation; lateral movement | Nonce-bound and time-limited VCs |
| 3 | Abusing Ephemeral Agents | Rapidly invoking/spawning multiple agents | **L3 (Agent Frameworks)** – Agent | Denial of service | Implementing rate limiting, detection of |

| | | | | | |
|---|---|---|---|---|---|
| | | that are short-lived to exhaust resources or mask malicious activity | instantiation logic **L4 (Deployment & Infrastructure)** – Resource allocation controls **L7 (Agent Ecosystem)** – Marketplace and discovery abuse | | anomalous behaviour and thus revocation |
| 4 | Escalation of Delegations | Chaining of delegations to obtain escalated privilege across agents | **L3 (Agent Frameworks)** – Delegation model **L6 (Security & Compliance)** – Policy enforcement **L7 (Agent Ecosystem)** – Cross-agent trust boundaries | Escalated privilege | Scoping of delegations, limiting re-delegation chains |
| 5 | Prompt Injection | Manipulation of prompts to foundation models to cause malicious/unwanted agent behavior | **L1 (Foundation Models)** – Prompt handling robustness **L3 (Agent Frameworks)** – Input sanitization **L6 (Security & Compliance)** – Prompt constraints and policy enforcement | Output corruption; RCE; Sensitive Data Exposure | Input Validation, prompt moderation/guardrails, |
| 6 | Data Poisoning | Ingesting malicious/unwanted/corrupt data into training dataset | **L2 (Data Operations)** – Data ingestion pipelines **L6 (Security & Compliance)** – Data governance | Output corruption | Verification of data/dataset integrity, anomaly detection in dataset pipeline |
| 7 | Model Inversion | Extracting model intellectual | **L1 (Foundation Models)** – Model | Theft of intellectual | Encryption at rest, |

| | | | | | |
|---|---|---|---|---|---|
| | | property or secrets | storage security **L4 (Deployment & Infrastructure)** – Storage and access controls **L6 (Security & Compliance)** – Data protection policies | property; theft of secrets/API keys | implementing access controls on APIs with model retrieval capabilities |

*Table 5: MAESTRO Security Threat Assessment for Multi-Agent Systems*

## 7.3 Cross-Layer Threats Affecting the IAM Framework

Including supply chain attacks on IAM components, privilege escalation across IAM layers, and goal misalignment leading to IAM misuse, all requiring defense-in-depth and continuous monitoring.

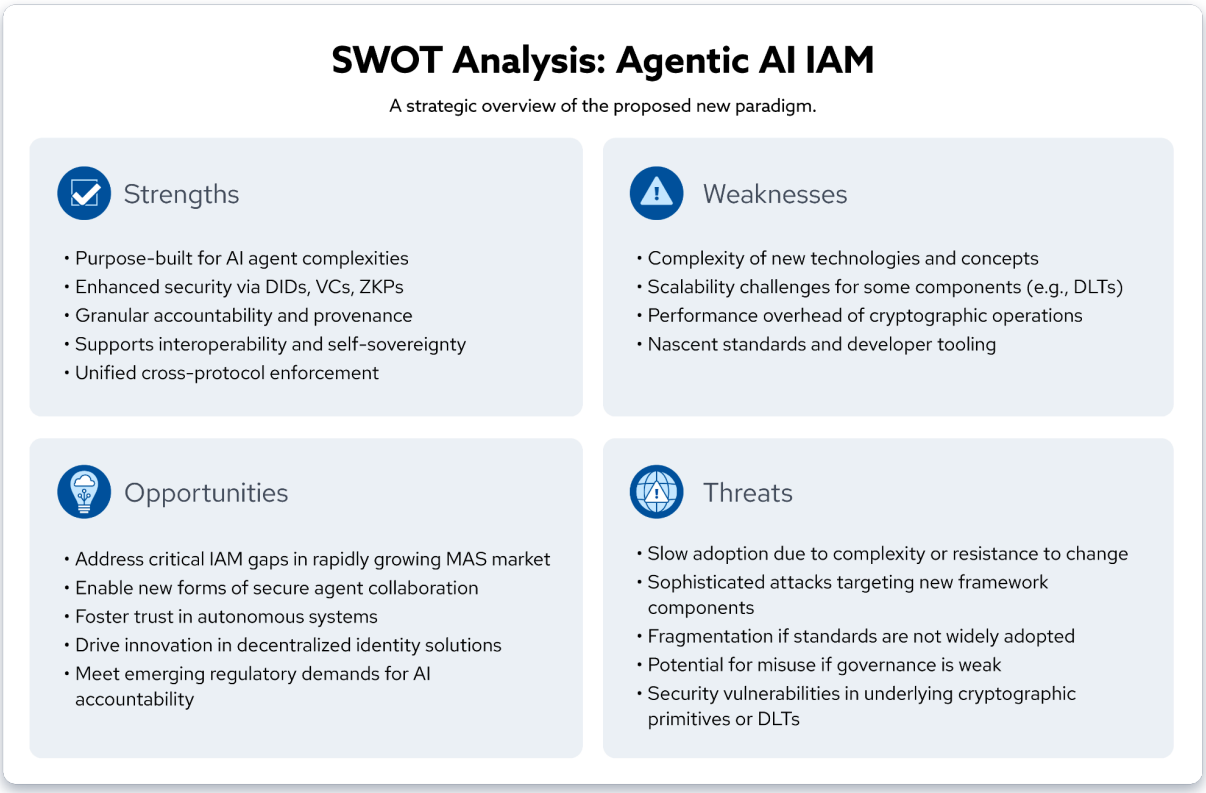


*Figure 8: SWOT Analysis: Agentic AI IAM*

## 7.4 Applying Zero Trust to Agentic AI IAM Framework

Applying Zero Trust to Agentic AI IAM Framework brings essential security, governance, and accountability benefits especially given the autonomous decision making, non-deterministic behavior, and scale of AI agents. Implemented and tested security controls that are preventative, detective, and corrective form the basis of Zero Trust. These fundamentals are critical to the success of a Zero Trust implementation:

- Concept of least-privilege access
- Separation of duties
- Segmentation/micro-segmentation
- Logging and monitoring
- Configuration drift remediation
- Assume breach
- Dynamic and adaptive security policy enforcement

## 7.5 Enterprise use cases with MAESTRO Framework:

| Use Case | Description |
|---|---|
| Agentic Red Teaming | Simulate offensive campaigns targeting LLM agents and tools |
| Security Chaos Engineering | Inject failure modes and threat vectors into agentic orchestration flows |
| Agent Trust Boundary Validation | Validate RBAC/ABAC scopes in dynamic multi-agent scenarios |
| Data Leakage Risk Analysis | Identify unintended propagation of sensitive data or vector embeddings |
| LLM Plugin & Tooling Risk Assessment | Model lateral movement through 3rd-party tools and open APIs |
| AI Identity Abuse Simulation | Test unauthorized identity impersonation or token misuse via agents |

*Table 6: Enterprise Security Testing Use Cases Using MAESTRO Framework*

# 8. Innovative Contributions of this Framework

The proposed framework represents a significant departure from traditional approaches, offering a collection of synergistic innovations specifically designed for the unique challenges of autonomous Multi-Agent Systems (MAS).

These contributions are not isolated features but form part of a re-conceptualization of agent identity, integrating advanced cryptographic techniques and a novel architectural design for dynamic control, all within a holistic, lifecycle-aware approach to managing AI agents as first-class digital citizens.

- The foremost contribution is the articulation of a **comprehensive, end-to-end IAM framework purpose-built for the agentic paradigm**.
    - This moves beyond merely adapting human-centric or simplistic machine and NHI(Non Human Identity) IAM protocols, which often prove inadequate for the complexities of autonomous, interacting agent swarms.
    - Instead, our framework cohesively integrates identity issuance, rich credentialing, capability-aware discovery, dynamic access control, and a novel cross-protocol enforcement layer into a unified conceptual model.
    - It addresses the entire lifecycle of an agent,from its "birth" through its operational interactions to its eventual decommissioning,recognizing the deep interdependencies between these stages.
    - Existing IAM solutions typically focus on narrower problems, struggling with identities that spawn others, dynamically change roles, or require fine-grained, context-sensitive authorization at massive scale. This framework's systemic integration is designed to address these fundamental gaps.

- Central to this is a **redefinition of Agent Identity, making it rich, dynamic, and verifiably secure**.
    - We shift away from simplistic identifiers like API keys towards identities anchored by cryptographically secure Decentralized Identifiers (DIDs).
    - This DID-anchored identity is not static; it is an extensible digital representation augmented by Verifiable Credentials (VCs) that attest to an agent's attributes, capabilities, compliance status, roles, and provenance.
    - The dynamism is crucial, as AI agents evolve, their models update, capabilities expand, and compliance needs re-attestation.
    - A rich, verifiable identity containing fields like scopeOfBehavior, toolset (which can include DIDs of authorized tools), modelHash, and VCs for training data or compliance, allows for far more nuanced trust and authorization.
    - The use of DIDs provides self-sovereignty and interoperability, essential for open MAS, while VCs offer a standardized, vendor-neutral way to make diverse claims.

    - Furthermore, Zero-Knowledge Proofs (ZKPs) enable agents to selectively and privately present these verifiable claims, a significant advancement over the limited flexibility and privacy of traditional certificate extensions.

- Building on this rich identity, the framework introduces **capability-centric discovery and more granular access control**.
    - An integrated Agent Naming Service (ANS) facilitates secure discovery, allowing agents to find others not just by name but by the specific functions or attested capabilities they offer.
    - This is a critical distinction from traditional service discovery, which may locate an endpoint but doesn't inherently verify the target's attested abilities.
    - Our approach directly links discovery to verifiable identity attributes.
    - Authorization decisions thereby become more intelligent, considering not just "who" is making a request, but fundamentally "what is this agent verifiably capable and authorized to do, with which specific tools, and under what attested conditions?"
    - By making an agent's authorized toolset and scopeOfBehavior verifiable parts of its identity, the system can enforce the principle of least function, significantly limiting the blast radius of a compromised or misbehaving agent.
    - The framework introduces Context-Based Access Control which enables dynamic access decisions based on real-time environmental, behavioral, and task context moving beyond static roles or attributes allowing enforcement policies to adapt to an agent's current state and conditions.

- A cornerstone innovation is the **Unified Cross-Protocol Global Session Management and Policy Enforcement Architecture**.
    - This Layer 4 uniquely addresses the challenge of maintaining consistent security posture in heterogeneous MAS where agents use diverse communication protocols.
    - In such environments, a critical security gap is the inability to propagate vital IAM state changes,like a global session termination, a master DID revocation, or a sudden capability downgrade,instantaneously and uniformly across all interaction points.
    - This layer acts as a "security and session management backplane," ensuring that a policy decision or revocation, once made, is effectively and immediately enforced wherever an agent might interact, regardless of the underlying transport.
    - This real-time, cross-protocol consistency is fundamental for operationalizing robust security.

- The framework also achieves a **pragmatic fusion of self-sovereignty with enforceable governance**.
    - While DIDs and agent-controlled VCs empower agents and their controllers with greater control over their core identity data, this self-sovereignty is balanced with mechanisms for practical governance.
    - This means that while an agent can present its self-managed identity, these credentials can be verified against established trust frameworks, such as lists of accredited VC issuers for specific roles or compliance attestations.

    - The Session Authority retains the ability to enforce global revocations or policy overrides based on enterprise risk decisions, even if the agent "controls" its DID.
    - This balance is vital for adoption in real-world systems that require clear lines of accountability and cannot operate solely on peer-to-peer trust.

- Finally, the framework provides **intrinsic support for fine-grained accountability and verifiable provenance**.

    - Cryptographic verifiability is embedded at multiple levels:
        - for identities via DIDs and their keys
        - for claims about agents via VCs and issuer signatures
        - for agent actions using the agent's DID-associated private key.
    - The Agent ID structure itself is designed to encapsulate or link to detailed provenance information,such as its creator, constituent models, software dependencies (potentially with their own DIDs), and VCs attesting to training data or safety audits.
    - As AI agents are entrusted with increasingly impactful decisions, the ability to irrefutably determine "who (which agent instance) did what, when, why, with what authority, and based on what information/capabilities" becomes critical.
    - This moves beyond basic logging to establish a cryptographically verifiable audit trail, essential for forensics, dispute resolution, and building societal trust in autonomous systems, directly addressing the "audit trail ambiguity" prevalent in current systems and providing a much stronger basis for non-repudiation.

**To measure the success implementation of the innovation, the following Key Performance Indicators (KPIs) can be considered:**

**Successful Agent Authentication Rate:** Percentage of successful agent authentications compared to attempted authentications within a defined time frame. A high rate indicates the framework's ability to consistently verify agent identities.

**Authorization Latency:** Average time taken to process and grant or deny an access request from an agent. Low latency ensures smooth agent operations and prevents bottlenecks.
Policy Enforcement Accuracy: Percentage of access requests that are correctly authorized or denied based on defined policies. A high accuracy indicates effective policy management and enforcement.

**Revocation Time:** Time taken to revoke access for a compromised or decommissioned agent across all systems it interacts with. Short revocation times minimize potential damage from compromised agents.

**Audit Log Integrity:** Percentage of successfully recorded and verifiable audit logs of agent activities. High integrity ensures accountability and traceability of agent actions.

**Anomaly Detection Rate:** Number of security anomalies or policy violations detected by the system within a specific time frame. This indicates the system's ability to identify deviations from expected agent behavior.

**Incident Response Time:** Time taken to detect, isolate, and mitigate a security incident involving a compromised agent. Short response times limit potential damage and disruptions.

**Agent Discovery Success Rate:** Percentage of successful agent discoveries through the Agent Naming Service (ANS) compared to attempted discoveries. This indicates the effectiveness of the discovery mechanism.

**Downtime due to IAM Issues:** Measure the total downtime or disruptions caused by issues with the Agentic AI IAM framework. Ideally, this KPI should be minimal or zero.

# 9. Discussion and Future Work

As future work, we have identified the following.

- **Scalability, Performance, and Efficiency:**
    - **The Challenge:** Several components within the proposed architecture, particularly those involving Distributed Ledger Technologies (DLTs) for DID registration and Verifiable Credential (VC) status management, or the Session State Synchronizer (SSS) which must track potentially millions of active agent sessions, face significant scalability and performance hurdles. The cryptographic operations inherent in DIDs, VCs, and Zero-Knowledge Proofs (ZKPs), while providing security, can also introduce computational overhead for resource-constrained agents or high-throughput systems.
    - **Future Work:**
        - **Benchmarking and Optimization:** Rigorous, large-scale benchmarking of different DLT solutions, consensus mechanisms, and distributed databases (for the SSS) under realistic MAS load conditions is essential. This includes measuring transaction throughput, latency for identity operations (registration, resolution, revocation), and query speeds.
        - **Efficient Cryptography:** Continued research into more lightweight and efficient ZKP schemes (e.g., optimizing proving and verification times, reducing proof sizes), more compact VC formats, and faster signature algorithms suitable for agentic environments.
        - **Caching and Resolution Strategies:** Developing sophisticated caching mechanisms for DID Documents and VC public keys, while ensuring timely propagation of revocation information, is crucial. Exploring hybrid resolution mechanisms that combine decentralized trust anchors with localized, high-performance caches.
        - **Hardware Acceleration:** Investigating the role of hardware acceleration (e.g., trusted execution environments, cryptographic co-processors) within agents or IAM infrastructure nodes to offload intensive cryptographic computations.

- **Standardization and Interoperability:**
    - **The Challenge:** The true power of a global Agentic AI IAM framework lies in its interoperability. Without widely adopted standards for how Agent IDs are structured, how capabilities are defined and attested in VCs, how ZKPs are constructed for common proofs, or how ANS queries are formatted, the ecosystem risks fragmentation into incompatible identity silos.
    - **Future Work:**
        - **Active Standards Development:** Proactive engagement with and contributions to standards organizations like the World Wide Web Consortium (W3C) for DIDs and VCs, the Internet Engineering Task Force (IETF) for potential ANS protocols or secure communication standards, the Decentralized Identity Foundation (DIF), and the Trust over IP (ToIP) Foundation.
        - **Agent-Specific Profiles:** Developing standardized VC profiles specifically for AI agents, covering common attributes like model_type, training_data_provenance, tool_authorization, ethical compliance attestations, and scope_of_behavior.
        - **Common Ontologies:** Creation of shared ontologies and vocabularies for describing agent capabilities, functions, and interaction patterns to ensure semantic interoperability when agents discover and attempt to use each other's services.
        - **Reference Implementations and Conformance Suites:** Building open-source reference implementations of key framework components (e.g., Agent ID SDKs, an ANS resolver, a basic Session Authority) and developing conformance testing suites to validate interoperability between different vendor and community implementations.
        - **Formalization of Model Context Protocols (MCPs):** Standardize the structure and exchange format of MCPs to enable agents to communicate intent, operational context, constraints, and task specifications in a verifiable and interoperable manner. As agent capabilities evolve, MCPs must support explicit versioning, extensibility, and schema negotiation to ensure forward compatibility and safe coordination across complex agent ecosystems.
- **Governance Models, Trust Frameworks, and Legal Considerations:**
    - **The Challenge:** Establishing and managing governance for a potentially global, decentralized, or federated IAM infrastructure is a monumental task. This includes defining who can issue authoritative VCs (e.g., for legal identity or compliance), how disputes over DIDs or ANS names are resolved, and how liability is attributed in complex MAS interactions. The evolving legal and regulatory landscape for AI also presents a moving target.
    - **Future Work:**
        - **Multi-Stakeholder Governance Research:** Investigating and prototyping various governance models (e.g., foundation-led, consortia-based,

DAO-controlled) for different components of the IAM framework, particularly for VC issuer accreditation and ANS root zone management.
        - **Trust Assurance Levels:** Defining clear trust frameworks with varying levels of assurance for Agent IDs and VCs, allowing verifiers to make risk-based decisions based on the rigor of the identity proofing and credential issuance processes.
        - **Legal and Regulatory Analysis:** Continuous analysis of emerging AI regulations (e.g., EU AI Act, national AI strategies) and data protection laws (e.g., GDPR) to ensure the IAM framework can support compliance. Developing guidance on topics like the legal standing of an agent's digital signature, cross-border data flows of VCs, and the "right to be forgotten" for Agent IDs.
        - **Dispute Resolution Mechanisms:** Designing fair and efficient mechanisms for resolving disputes related to identity claims, VC validity, or malicious agent behavior attributed via the IAM framework.
        - **Security Controls Specific to AI Agents:** Developing new controls and adapting existing controls for Agentic AI systems, including the establishment of baseline security controls, continuous monitoring controls, and the automated enforcement of controls. These controls should also align with established industry frameworks such as the Cloud Controls Matrix (CCM), ISO/IEC 2700X, NIST SP800-53, and others to ensure consistency, auditability, and integration into broader risk management programs.
        - Autonomous Identity Agents for Just-In-Time (JIT) Access Decisions.
        - Multi-Agent Systems for Decentralized Trust Brokerage
        - Explainable and Auditable Agent Decisions in IAM
        - Proactive Access Threat Modeling via Simulated Agent Behavior
        - Interfacing Agentic IAM with AI-Augmented SOCs (Security Operations Centers)

- **Enhanced Security and Privacy in Practice:**

    - **The Challenge:** While the framework incorporates strong security primitives, sophisticated adversaries will inevitably seek to exploit implementation weaknesses, social engineering aspects, or unforeseen interaction effects between components. Maintaining agent and user privacy in the face of increasingly rich identity data is also paramount.

    - **Future Work:**

        - **Formal Security Modeling and Verification:** Applying formal methods to mathematically prove the security properties of critical protocols within the framework, such as the runtime ID assumption protocol or the cross-protocol revocation mechanism.
        - **Agent-Specific Threat Intelligence:** Developing and sharing threat intelligence specifically focused on attacks against agentic systems and their IAM components (e.g., novel ways to poison training data to acquire VCs, exploiting ZKP library vulnerabilities).
        - **Advanced Privacy-Enhancing Technologies (PETs):** Exploring the integration of more advanced PETs beyond basic ZKPs, such as homomorphic encryption for computations on encrypted agent VCs, or attribute-based

encryption for fine-grained data access control directly tied to verifiable attributes.
        - **Secure Key Management for Autonomous Agents:** Researching best practices and developing robust solutions for secure private key generation, storage, usage, and rotation within autonomous agents, especially those operating in untrusted or resource-constrained environments. This includes exploring multi-party computation (MPC) for key management.
        - **Resilience Against Quantum Threats:** Proactively investigating and planning for the transition to quantum-resistant cryptographic algorithms for DIDs, VCs, and digital signatures.
        - **Tabletop Exercises for Agentic Incident Response:** Designing and conducting tabletop exercises focused on scenarios such as compromised agent identities, unauthorized delegation, policy conflict resolution, or mass revocation events. These exercises help identify operational challenges and test the effectiveness of IAM controls, policy enforcement, and incident response readiness.

- **User Experience (UX), Developer Tooling, and Adoption Pathways:**
    - **The Challenge:** For this framework to be adopted, it must be usable by both end-users (who may act as controllers for their personal agents) and developers building and deploying AI agents. Complexity in managing DIDs, VCs, and policies can be a significant barrier.
    - **Future Work:**
        - **Developer-Friendly SDKs and Libraries:** Creating intuitive and well-documented Software Development Kits (SDKs) for various programming languages that simplify Agent ID creation, VC management (issuance, holding, presentation), ZKP generation, and interaction with the IAM framework's services (ANS, PDP, SA).
        - **Management UIs and Dashboards:** Developing user interfaces for agent controllers to manage their agents' identities, review their VCs, set basic policies, and monitor their activity.
        - **"Secure by Default" Agent Architectures:** Promoting agent development patterns and templates that embed IAM best practices from the outset.
        - **Phased Adoption Strategies:** Defining clear pathways for organizations to incrementally adopt components of the framework, potentially integrating with their existing IAM systems first (e.g., using enterprise OIDC to bootstrap an agent controller's DID) before moving to more decentralized elements.
- **Ethical Considerations and Societal Impact Mitigation:**
    - **The Challenge:** The power of verifiable and persistent Agent IDs, while beneficial for security, also carries potential risks if misused for pervasive surveillance, biased decision-making (e.g., if VCs for "good behavior" are only available to certain types of agents), or creating new forms of digital divide.

- **Future Work:**
    - **Ethical Impact Assessments:** Establishing methodologies for conducting ethical impact assessments specifically for Agentic AI IAM deployments, considering fairness, accountability, transparency, and potential for misuse.
    - **Bias Detection and Mitigation in Credentialing:** Researching techniques to detect and mitigate biases in the issuance of VCs, particularly those related to agent capabilities, reputation, or compliance.
    - **Transparency and Explainability of IAM Decisions:** Ensuring that decisions made by the IAM framework (e.g., why an agent was denied access, why a VC was revoked) are explainable to the relevant stakeholders.
    - **Public Discourse and Inclusive Design:** Fostering broad public and interdisciplinary discussions involving ethicists, social scientists, policymakers, and diverse user groups to shape the development and deployment of Agentic AI IAM in a responsible manner. Ensuring that the framework is designed to be inclusive and does not inadvertently disadvantage certain groups or types of agents (e.g., open-source or community-developed agents).

The journey to a fully realized and globally functional Agentic AI IAM framework is an ambitious one. It necessitates a collaborative, iterative approach, blending cutting-edge research with pragmatic engineering and a deep understanding of the evolving societal context of AI. Addressing these future work areas will be critical to transforming the vision presented in this paper into a resilient, trustworthy, and enabling infrastructure for the future of AI.

# 10. References


Rosenbush, Steven. "AI Agents Face One Last, Big Obstacle." *The Wall Street Journal*, 17 May 2025

Chan, A., et al. (2024b). *IDs for AI Systems*. arXiv preprint arXiv:2406.12137.

European Parliament and Council. (2023). *Regulation of the European Parliament and of the Council on Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts*.

Gabriel, I., et al. (2024). *The Ethics of Advanced AI Assistants*. arXiv preprint arXiv:2404.16244.

Goldwasser, S., Micali, S., & Rackoff, C. (1989). The knowledge complexity of interactive proof systems. *SIAM Journal on Computing, 18*(1), 186-208.

Hardt, D. (Ed.). (2012). *The OAuth 2.0 Authorization Framework* (RFC 6749). Internet Engineering Task Force. https://doi.org/10.17487/RFC6749

Hardt, D., Parecki, A., & Lodderstedt, T. (2025, May 28). The OAuth 2.1 Authorization Framework (Internet Draft No. draft-ietf-oauth-v2-1-13). IETF. Retrieved from https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-13

Kindervag, J. (2010). *Build Security Into Your Network's DNA: The Zero Trust Network Architecture*. Forrester Research.

Mockapetris, P. (1987). *Domain names - implementation and specification* (STD 13, RFC 1035). Internet Engineering Task Force. https://doi.org/10.17487/RFC1035

OASIS. (2005). *Security Assertion Markup Language (SAML) V2.0 Errata*. OASIS Standard.

OpenID Foundation. (2014). *OpenID Connect Core 1.0 incorporating errata set 1*. OpenID Foundation Standards. Retrieved [Date of Retrieval] from https://openid.net/specs/openid-connect-core-1_0.html

Sporny, M., et al. (Eds.). (2024, May). *Verifiable Credentials Data Model v2.0*. W3C Working Draft. World Wide Web Consortium. Retrieved [Date of Retrieval] from https://www.w3.org/TR/vc-data-model-2.0/

World Wide Web Consortium (W3C). (2021, November 19). *Verifiable Credentials Data Model v1.0*. W3C Recommendation. Retrieved [Date of Retrieval] from https://www.w3.org/TR/vc-data-model/

World Wide Web Consortium (W3C). (2022, July 19). *Decentralized Identifiers (DIDs) v1.0*. W3C Recommendation. Retrieved [Date of Retrieval] from https://www.w3.org/TR/did-core/

Huang, K., Narajala, V. S., Habler, I., & Sheriff, A. (2025, May). *Agent Name Service (ANS): A universal directory for secure AI agent discovery and interoperability* (Internet-Draft No. draft-narajala-ans-00). Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-narajala-ans/

Huang, K. (2025a, March 11). *Agentic AI identity management approach*. Cloud Security Alliance. https://cloudsecurityalliance.org/blog/2025/03/11/agentic-ai-identity-management-approach

Huang, K. (2025b, February 6). *Agentic AI Threat Modeling Framework: MAESTRO*. Cloud Security Alliance Blog.

Huang, K (2025c), Agentic AI DID SDK: Decentralized Identifiers and Zero-Knowledge Proofs, https://github.com/kenhuangus/agent-id-sdk

Sheriff,A (2025d), Agentic AI DID SDK: Decentralized Identifiers and Zero-Knowledge Proofs, https://github.com/akramIOT/Agentic-IAM/

Huang, K., Sheriff, A., Narajala, V. S., & Habler, I. (2025). Agent Capability Negotiation and Binding Protocol (ACNBP). arXiv preprint arXiv:2506.13590. https://doi.org/10.48550/arXiv.2506.13590

Huang, J., Huang, K., Hughes, C. (2025). AI Agents in Offensive Security. In: Huang, K. (eds) Agentic AI. Progress in IS. Springer, Cham. https://doi.org/10.1007/978-3-031-90026-6_6

Huang, J., Huang, K., Hughes, C. (2025). AI Agents in Defensive Security. In: Huang, K. (eds) Agentic AI. Progress in IS. Springer, Cham. https://doi.org/10.1007/978-3-031-90026-6_7

Huang, J., Huang, K., Jackson, K., Hughes, C. (2025). AI Agent Safety and Security Considerations. In: Huang, K. (eds) Agentic AI. Progress in IS. Springer, Cham. https://doi.org/10.1007/978-3-031-90026-6_12

OWASP(2025), OWASP Top 10 Non-Human Identities Risks - 2025, https://owasp.org/www-project-non-human-identities-top-10/2025/top-10-2025/

Cloud Security Alliance(2024), The State of Non-Human Identity Security, https://cloudsecurityalliance.org/artifacts/state-of-non-human-identity-security-survey-report