

Who is the ‘I’ in AI? Establishing Decentralized Identity and Accountability for Autonomous Agents

Ahammod Munim Ramil*, Khalid Al Kafi*, M. A. Abdullahis Sany*, Muktadirul Islam Nibir*,
Md Yeasin Ali*, Mohammad Zavid Parvez†, and Md Sadek Ferdous*

**Department of Computer Science and Engineering*

BRAC University, Dhaka, Bangladesh

Emails: {ahammod.munim.ramil, khalid.al.kafi, ma.abdullahis.sany, muktadirul.islam.nibir}@g.bracu.ac.bd,
{yeasin.ali, sadek.ferdous}@bracu.ac.bd

†School of Computing, Mathematics and Engineering

Charles Sturt University, Bathurst, New South Wales, Australia

Email: zavidparvez@gmail.com

Abstract—The rapid proliferation of autonomous AI agents in decentralized ecosystems challenges existing identity paradigms. While human identity is traditionally anchored in biological and/or self-asserted attributes and government credentials, AI agents lack a standardized definition of selfhood, creating critical gaps in trust and accountability. This paper introduces a novel framework for establishing Decentralized AI Identity rooted in Self-sovereign identity (SSI) principles. We propose a formal model that clearly distinguishes human users and AI agents, enabling distinct verification logic for each class while maintaining a unified identity calculus. The central to our contribution is the ABCD Framework, a multi-layered approach to agent identification and verification: (A) Algorithmic Fingerprint, which binds the identity to static model architectures and provenance; (B) Behavioural Fingerprint, which verifies dynamic performance metrics over time; (C) Cryptographic Identity, which anchors the agent via Decentralized Identifiers (DIDs) and (D) Delegation, which cryptographically binds agent actions to respective human accountability. By orthogonalizing these dimensions, we model how systems can enforce granular trust policies ranging from simple authentication to complex liability assignment, thereby solving the black box identity problem in the autonomous economic activities of AI agents.

Index Terms—Decentralized Identity, Autonomous Agents, Blockchain, Accountability

I. INTRODUCTION

In contrast to the traditional generative models that are characterised as passive and request-response frameworks, Agentic AI systems are characterised by their ability to plan proactively, use advanced tools, and adapt to the environment [1], [2]. These systems act as autonomous principals rather than mere tools. They are dynamically sensitive to complex environments and reason about abstract objectives and organize progressively multi-step workflows to transform states without human intervention on a continuous basis [3], [4].

And now, imagine that in the future everybody has one or more AI agents, just like a digital assistant. Although the traditional concept of human identity is formed on the basis of physical embodiment and can be verified with the help of state-produced identification symbols (e.g., passports) or digital identification that can be verified (e.g., email), AI agents do not have any biological and legal basis [5]. Being computational

entities, AI agents have neither physical sovereignty nor natural legal personhood [5]. This ontological gap is one of the critical issues in Multi-Agent systems. Specifically, we lack a deterministic way that is able to distinguish individual agents among the thousands of structurally identical agents.

Although the autonomy of Agentic AI facilitates collective problem solving as well as efficiency [6], the basic detachment between actions and the short-term human intentions breaks down the current identity paradigms. The reason behind this undertaking is a latent ‘Identity Gap’ that is presently making decentralized agentic economies insecure. The existing systems have biological anchors (e.g., biometrics, KYC) or static infrastructure credentials (e.g., OpenID, OAuth tokens) [7], [8], that simply do not fit the nature of autonomous software. AI agents are temporary automated programs that can be forked and run at machine speed; they do not have a physical entity to be seen to, much as there is no legal entity to be held liable to. Since an agent may be conducting transactions of high value, or making very sensitive decisions, without a human observer, this creates an accountability vacuum because of the lack of a deterministic and cryptographically verifiable identity layer. Within such a vacuum, bad actors are able to run Sybil swarms of cloned or model-swapped agents to control markets, avoid reputation systems, and commit fraud with plausible deniability. Providing a clear and verifiable identity to algorithmic entities is thus not only an operational improvement, but a security requirement of Multi-Agent Systems (MAS) [9].

Recent proposals have attempted to address these challenges through social governance or operational orchestration. However, these frameworks predominantly treat agent identity as an extrinsic label attested by third parties or reputation scores. They lack the mechanisms to cryptographically bind an agent’s runtime execution to its static model architecture.

Our work addresses these specific architectural deficiencies by introducing a Self-sovereign Identity (SSI) [10] based AI identity framework, called the *ABCD Framework*, which orthogonalizes identity into four verifiable dimensions and addresses the black box problem in reputation-based models [11]: Algorithmic Fingerprinting, Behavioural Fingerprinting,

Cryptographic Identity and Delegated Liability. In particular, we make the following concrete, verifiable contributions:

- **A Formal, Domain-Aware Identity Model.** We introduce a formal domain model that strictly differentiates human principals from autonomous AI agents.
- **The ABCD Framework.** We present the ABCD Framework, a unified identity calculus that combines algorithmic provenance (code), behavioural integrity (performance), cryptographic anchoring (keys), and delegated liability (ownership).
- **Formal Security Guarantees.** We formally state and prove three security theorems, grounded in explicit cryptographic and systems assumptions: Behavioural Coherence and Tampering Detection, Sybil and Impersonation Resistance, and Temporal Bounded Authorization Enforcement.

II. BACKGROUND

Self-Sovereign Identity (SSI): SSI represents a paradigm shift in digital identity management that aims to empower users so that they can exert more control over their identity data [12]. There are three actors in an SSI ecosystem. An *Issuer* is an entity (e.g., a regulatory body or model auditor) that asserts cryptographically verifiable claims about a Subject in the form of a tamper-evident Verifiable Credentials (VCs). The *Holder* (Subject) is an entity that possesses and controls the credentials, typically stored within a secure digital wallet or agent framework. The *Verifier* is an entity that requests and receives VCs from the holder and validates them to authorize access or interactions to the holder. These interactions between these actors thus forms a trust triangle.

Decentralized Identifiers (DIDs): DIDs are a globally unique identifier standardized by the W3C to enable verifiable, self-sovereign digital identity [13] [14]. Unlike traditional identifiers (e.g., email addresses) issued by centralized authorities, DIDs are created from a public key and owned by the subject controlling the respective private key [14]. Operationally, a DID is a URI that resolves to a DID Document, a data structure containing the subject's public keys and service endpoints. This binding allows an entity to cryptographically prove ownership of its identity via digital signatures, decoupling authentication from centralized registries [15]. One might ask why agents cannot simply be assigned centralized identifiers such as emails, as in human accounts. Centralized issuance presupposes a trusted registrar and a persistent, singular endpoint. This is a poor fit for entities forked, cloned, or retired at machine speed across mutually distrusting platforms with no shared registrar; Definition 4's uniqueness requirement is therefore satisfied only by self-certifying DIDs.

Verifiable Credentials (VCs): VCs are portable, tamper-proof and cryptographically signed digital claims made about a subject [12] [13]. A credential is made of a set of individually verifiable claims. By leveraging VC, a subject can present the claim to a verifier when requested.

Verifiable Data Registry (VDR): A VDR is another important component that is used to store DIDs, DID Docs, and

credential schema and acts as a revocation registry. Generally, blockchain is used as a VDR because of its data immutability and decentralisation characteristics [16], enabling trustless credential validation without a central authority.

Multi-Agent Systems (MAS): MAS is the paradigm shift between monolithic computation to distributed intelligence [9]. A MAS is a computational system that consists of multiple interacting intelligent agents aimed at working together to meet individual or common goals [17]. These systems deal with complex problems by breaking them down into smaller manageable subtasks, each of which is delegated to a specific agent [9]. Unlike passive components, agents possess the inherent ability to learn from their environment and interactions, allowing them to make autonomous decisions to achieve their goals.

Autonomous AI Agents/Agentic AI: AI Agents or Agentic AI systems are application systems built on large LLMs to solve a particular problem or a series of problems using AI [6]. Autonomous AI agents are AI agents that can work autonomously, even without human intervention [6]. However, current identity systems are not equipped to verifiably differentiate AI agents from humans, which can create vulnerabilities such as impersonation, sybil attacks, and unregulated economic behaviour.

III. RELATED WORK

The problem of establishing trust in Multi-Agent Systems (MAS) is not new, but the emergence of autonomous economic agents (agents carrying financial operations, e.g., executing trades or transferring values, as distinct from purely advisory agents) has rendered existing paradigms insufficient. We categorize existing literature into three streams: general MAS security foundations, ethical governance frameworks, and zero-trust operational architectures.

Security Challenges in MAS: Early research into MAS security established the fundamental difficulties of coordination and distributed trust. Dorri et al. [9] provided a comprehensive taxonomy of MAS challenges, highlighting that while agents offered flexibility and efficiency, they introduced severe vulnerabilities regarding sociability and mobility. They identified that without a central trusted authority, the spread of falsified information by malicious agents could be easily propagated to influence decision making. Our work is based on the following foundational definitions of autonomy and sociability but expands the security model to consider the unique threats, like identity spoofing of AI agents in cyberspace.

Ethical Consensus and Governance Layers: The attempts that have been made recently have tried to impose ethical limitations on agent behavior. The LOKA Protocol [11] suggests the use of a Universal Agent Identity Layer (UAIL) together with a Decentralized Ethical Consensus Protocol (DECP) to bring the agents abreast with human values by adopting a voting system. However, LOKA fundamentally treats the agent as a black box voter within a DAO structure rather than a verifiable computational entity. Though it makes use of reputation

scores to authorize social behavior, it does not consider the Algorithmic Fingerprinting aspect (the "A" in our model) at all. Critically, LOKA provides no cryptographic mechanism to ensure that an agent's runtime execution matches its declared model architecture. Furthermore, LOKA attempts to resolve disputes via decentralized dispute resolution and consensus voting. In different applications, e.g. DeFi (Decentralized Finance) environments, a single malicious inference can drain a liquidity pool instantly. This reliance on post-hoc social arbitration is insufficient for real-time security. Our framework departs from this by requiring intrinsic cryptographic proofs of algorithmic provenance (Layer A) and delegated liability (Layer D) that pre-binds collateral to agent actions.

Zero-Trust Architectures for Agentic AI: The most operationally mature proposal to date is the Zero-Trust Identity Framework by Huang et al. [17]. They correctly identified that traditional OAuth/OIDC protocols were too coarse-grained for ephemeral, high-frequency agent workflows. Their contribution includes an Agent Name Service (ANS) [18] for discovery and a Unified Global Session Management layer that handles revocation across heterogeneous protocols. However, a critical security gap remains: they treated an agent's underlying model merely as static metadata (a `modelHash`) within a DID Document, relying entirely on external Trusted Issuers to attest to capabilities via Verifiable Credentials (VCs). This creates a dependency on third party certification instead of intrinsic cryptographic binding. It also fails to prove the internal architecture and inference integrity of the agent at runtime. Furthermore, their model takes human users and AI agents into one single class of participants who are differentiated only by scores of trust. This absence of any formal partitioning rules prevents the use of different verification logic such as requiring biological liveness of human versus cryptographic model proofs of agents which is critical for Sybil resistance in open economic protocols. In our work, we tackle this issue by orthogonalizing these dimensions in order to have a unified identity calculus which strictly separates and distinguishes biological principals from algorithmic agents.

Workload Attestation Standards: Industry standards such as SPIFFE/SPIRE [19] issue cryptographic workload identities (SVIDs) for cloud-native infrastructure, analogous in spirit to our Layer C. However, SPIFFE attests *infrastructure* identity (which node/container is running) rather than *model* identity (which weights are running) or *behavioural* conformance, and has no analogue to Layers A, B, or D. It is complementary to, not competing with, ABCD, and could serve as a transport-layer credential underneath Layer C.

Gap Analysis and Our Contribution: In summary, according to Table I, existing literature has a hard time finding autonomous economic agents due to their dependence on extrinsic trust architectures. For example, the LOKA Protocol [11] includes a Universal Agent Identity Layer (UAIL), however, it is essentially a black box about the runtime execution of an agent. Likewise, the Zero-Trust framework suggested by Huang et al. [17] is an overly dependent on external Trusted

Issuers in terms of capabilities and combines human and AI actors into a single session-management paradigm.

Our model stands out by mapping particular security vulnerabilities to a four-layer inherent defense. Where Huang et al. [17] use session tokens, our *Layer C: Cryptographic Identity* strictly ontologically separates humans and agents. Where LOKA [11] is based on social voting, our *Layer A: Algorithmic Fingerprint* offers intrinsic cryptographic code provenance. Existing frameworks effectively solve the problems of orchestration (Huang et al. [17]) and social alignment (LOKA) [11], but they fail to solve the problem of identity coherence. Currently, a malicious actor can register a valid DID (following Huang) and accrue high reputation (following LOKA), yet secretly swap the underlying model for a malicious variant at runtime. Moreover, we are the first to introduce *Layer B: Behavioural Fingerprint* to identify runtime inconsistencies with described models (to avoid model-swapping), and *Layer D: Delegation and Ownership* to create non-repudiable liability, which are either simply absent or are offloaded to dispute resolution in earlier literature. Concretely, replaying the AgentX/AgentY impersonation scenario of Section VIII against Huang et al.'s scheme [17] exposes the gap. Their `modelHash` is inert metadata attested only by an external Trusted Issuer, so an adversary who compromises that Issuer can re-attest a swapped model without triggering any runtime check. Whereas Theorem 1's coherence check is issuer-independent and flags the same swap from behaviour alone. Likewise, LOKA [11] has no analogue to Theorem 3. It can only down-vote an impersonator *after* the fact via its DECP, with no cryptographic rejection at the point of forgery.

IV. PROBLEM DEFINITION & THREAT MODEL

Formally, no identity framework satisfies all the following three conditions simultaneously:

- **Entity Differentiation:** The framework must strictly differentiate between human users (E^U) and autonomous agents (E^{AI}). It should also apply distinct verification protocols to each class rather than treating them as a unified set.
- **Identity Coherence:** An agent's cryptographic identity should all be intrinsically tied with their algorithmic state ($\Phi_{Alg}(x)$, defined later) and verifiable runtime behaviour ($\Phi_{Beh}(x, W)$, defined later).
- **Accountable Delegation:** Independent actions are to be held accountable to a responsible human principal through highly bounded authorization.

The absence of these properties still exposes decentralized ecosystems to the Sybil attacks, threats of runtime tampering and unaccountable damages.

A. System Model and Assumptions

The proposed system is assumed to be open and permissionless, i.e., there is no trusted intermediary between humans and agents. We limit our scope to agents that exceed a set cutoff limit of autonomy, so that actions in the state are carried out without certain human authorization. We assume

TABLE I
COMPARISON OF AGENT IDENTITY FRAMEWORKS MAPPING FEATURES TO THE ABCD LAYERS

Framework Layer (Feature)	Dorri et al. [9]	LOKA Protocol [11]	Huang et al. [17]	ABCD Framework (Ours) & Core Advantage
(C) Cryptographic Identity	Generic IDs	UAIL (DIDs & VCs)	DIDs + VCs + ANS	DID + Formal Partition: Prevents User-to-Agent impersonation via strict entity class separation.
(A) Algorithmic Fingerprint	Not addressed	Ethical Consensus (CEP)	Trusted Issuers (VCs)	Intrinsic Model Hash: Cryptographically binds identity directly to code, removing 3rd-party reliance.
(B) Behavioural Fingerprint	Fault Detection	Post-hoc Voting	Session Management	Runtime Coherence: Detects malicious model-swapping by checking runtime metrics against Layer A.
(D) Delegation & Liability	Task Allocation	Dispute Resolution	Global Revocation	Cryptographic Binding: Enforces time-bounded authorization tying agent actions to a human principal.

that adversaries possess perfect visibility of a network, are able to register valid DIDs and fork models at negligible cost. Normal cryptographic primitives are thought to be safe. Trusted execution hardware (TEEs) is not covered to guarantee wide compatibility. This threat model is scoped to the agent and its direct cryptographic/economic interactions (registration, delegation, transaction execution); enterprise perimeter threats (e.g. internal network segmentation, employee credential compromise) are treated as orthogonal and complementary to this identity layer, and hence, not covered here.

B. Threat Model

The modeled threats are based on the STRIDE [20] framework, mapped to the proposed ABCD framework in the following:

- ST1. Spoofing (Sybil Attacks):** An adversary orchestrates many distinct, individually-valid agent identities (a Sybil swarm), or impersonates a single registered agent by presenting credentials it does not control.
- ST2. Tampering (Model Swapping):** An agent envelops an altered or malicious model variant at runtime.
- ST3. Repudiation (Liability Evasion):** A human owner does not claim responsibility for the actions of an autonomous agent.
- ST4. Information Disclosure:** The proprietary model weights or architecture information leaked.
- ST5. Denial of Service (DoS):** Malicious actors may create multiple AI agents, launching a form of Sybil attack, which may exhaust system resources (e.g., storage, bandwidth), rendering the services unavailable to other legitimate users.
- ST6. Elevation of Privilege:** An agent exceeds the scope of its delegated authority by invoking a permission it was not granted, spending beyond an assigned budget, or continuing to act beyond its legitimate time frame.

V. FORMAL AI IDENTITY MODEL

In this section, we establish a formal AI Identity model for AI agents, grounded on the Digital Identity Model (DIM) [21]. Our model uses the set-theoretic calculus of the DIM and in particular, its formalization of entities, attributes and partial identities. We have extended the DIM model by introducing formal logic relevant for the orthogonal verification layers for the ABCD framework to ensure accountability for AI agents.

To maintain clarity, we use a subscript in a notation to denote the domain and a superscript to specify the entity. For instance, E_d^{AI} represents the set of AI entities in domain d . Where the original DIM [21] targeted identity correlation for a single human user across centralized providers, our extension targets a materially different problem, correlating a non-human, forkable entity across mutually distrusting, permissionless domains with no shared provider.

A. Entities and Domains

This section formalizes the mathematical primitives underlying the ABCD framework through a domain-aware entity model (Definition 1) that distinguishes human users from AI agents, classifies attributes by their identifying capability (Definition 2), and builds identity constructs that support selective disclosure (Definition 3).

Entities, Attributes, and Valuation: Identity statements are meaningful only with respect to a scope of interpretation. We call such a scope a *domain*. Within each domain we distinguish human users from AI agents while treating both uniformly as entities. This separation allows us to reason about accountability and automation without breaking the uniform calculus of identity. We do not suggest a redefinition of human digital identity or building the internet again. Human entities still authenticate to the existing, mature Web2/Web3 infrastructure (e.g., OAuth, or simple DIDs). The framework merely binds these already existing human identifiers to the new, four-layered identity of the autonomous agent cryptographically.

Definition 1 (Entities and Domains). Let D be the set of domains. For each $d \in D$, the entities of d are

$$E_d = E_d^u \cup E_d^{AI}, \quad (1)$$

where E_d^u are user entities and E_d^{AI} are AI entities, with $E_d^u \cap E_d^{AI} = \emptyset$.

Take the case of a DeFi system in which human traders trade with autonomous trading bots. Both participate in the same market, yet their identity requirements differ fundamentally. Humans bring biometric attributes, government credentials, and legal personhood. AI agents bring computational fingerprints, behavioural traces, and delegated authority. The domain is the DeFi platform itself but the scope within which identity claims have meaning.

TABLE II
ATTRIBUTE SETS PER ABCD LAYER

Layer Attribute Set	Purpose
A_A {modelHash, architectureID, fineTuneHash, cfgSnap, versionTag, issuerDID, issuerSignature}	Static algorithmic provenance
A_B {interactionPattern, ethicalProfile, decisionTraces, latency95, refusalRate, riskProfile, benchmarkScore, taskProfile}	Dynamic runtime behaviour
A_C {DID, publicKey, keyFingerprint}	Cryptographic identity
A_D {ownerDID, ownershipVC, delegationCert, delegationWindow}	Delegation & ownership

It is important to highlight that we do not redefine the human digital identity. Human entities still can authenticate using existing, mature Web2/Web3 identifiers (e.g., email or simple DIDs). The framework merely binds these already existing human identifiers to the new, four-layered identity of an AI agent cryptographically. Human users are added to the formal model only to be the responsible principals (delegators/owners) that the Layer D needs.

To describe entities we fix, per domain, a vocabulary of attributes together with the set of values they may take. For the AI-specific domain it is convenient to separate static provenance, dynamic behaviour, cryptographic bindings, and governance attributes; this mirrors verification steps used later.

Definition 2 (Attributes and Values). For each $d \in D$, let A_d^{AI} be the set of attributes and AV_d^{AI} the set of attribute values for AI agents within that domain. In respect to the AI ecosystem we partition

$$A_d = \mathbf{A}^A \cup \mathbf{A}^B \cup \mathbf{A}^C \cup \mathbf{A}^D. \quad (2)$$

Here, $\mathbf{A}^A$ represents the algorithmic attributes, $\mathbf{A}^B$ represents the behavioral attributes, $\mathbf{A}^C$ provides the cryptographic proof of identity, $\mathbf{A}^D$ links the AI agent to human accountability and governance rules representing the delegation and ownership attributes. We define each of the attributes sets in Table II. It is clear from Table II that attributes are the descriptors that give meaning to an identity.

The attributes of A^A represent the static computational essence of the AI agent, e.g. the cryptographic digest of its neural network weights, a human-readable description of its architecture, modifications applied through fine-tuning, a tamper-evident record of updates, or how the agent actually behaves in operation, providing evidence that static specifications translate to expected runtime characteristics. For example, a content moderation agent might have `modelHash = sha256:a1b...`, `architectureID = "BERT-large-uncased-toxicity-v2"`, indicating its base architecture and specialization. Similarly, for A^B , a trading AI agent might exhibit `latency95 = 150ms` for decision-making, `refusalRate = 0.02` for transactions violating risk parameters, demonstrating adherence to performance and safety constraints.

A^C consists of the Agent DID which acts as the unique identity handle. Public keys are resolved through the DID Document, and `keyFingerPrint` is the finger print of the key. Finally, A^D includes attributes that enable delegation and ownership verification, such as `ownerDID`, `ownershipVC`, and `delegationCert`. The semantics of ownership and delegation are formally specified in Definition 14, ensuring bounded authorization consistent with wallet constraints and the governance logic.

Next, we provide a few examples:

- A^A (Algorithmic): `modelHash = 0x3f2...`, `architecture = "Transformer-6B"`, `versionTag = "v2.1"`
- A^B (Behavioural): `latency95 = 220ms`, `refusalRate = 0.03`, `avgTradeSize = $5000`
- A^C (Credential): `DID = "did:ethr:0x742d35..."`, `publicKey = "0x04a5b..."`
- A^D (Delegation & Ownership): `ownerDID = "did:ethr:Alice"`, `maxTradeSize = $50000`, `delegationWindow = "2025-01-01-2025-12-31"`

This partition allows for modular verification, which enables auditors to independently confirm algorithmic provenance (A^A) without having to examine governance or delegation rules (A^D), credential integrity (A^C), or behavioural compliance (A^B).

B. Identity Calculus

Attributes are useful only when instantiated for specific entities. Not every AI agent possesses every attribute. For example, one agent *AgentX* may have a `modelHash` attribute (capturing its architecture version), but another agent *AgentY* may not. Similarly, some agents might record a `traceLogHash` for behavioural monitoring, while others omit it. Even when an attribute is applicable, an AI agent may withhold its value for privacy, security, or policy reasons—for instance, an agent might refuse to disclose its `apiKey` or internal `configHash`. That is why we model the valuation function ($atEntToVal_d$) as a partial map as defined in Equation 3. Hence, This function captures both non-applicability and non-disclosure through partiality.

Definition 3 (Valuation Function). For any $d \in D$,

$$atEntToVal_d : A_d \times E_d \rightharpoonup AV_d. \quad (3)$$

Undefinedness covers both non-applicability (an AI has no `passportNo`) and non-disclosure (a user withholds email). This minimal device lets identities be sparse without breaking subsequent definitions. From the following example, *AgentX* might have a value `modelHash` while the value of `configHash` for *AgentY* is undefined in an AI domain d_{AI} .

$$atEntToVal_{d_{AI}}(\text{modelHash}, AgentX) = 0x3f2a9c84b7... \\ atEntToVal_{d_{AI}}(\text{configHash}, AgentY) \text{ is undefined}$$

Attribute Classification:

A domain needs a unique naming attribute, called *Identifier*, which is used to uniquely identify an entity within that domain.

Other attributes may still identify in weaker senses, or may be undefined, however, an identifier must have a value for each entity. An identifier can be defined in the following way:

Definition 4 (Identifier). An *identifier* with respect to domain $d \in D$ is a function

$$\text{identifier} : D \rightarrow A^C$$

such that for any $d \in D$, if $i = \text{identifier}(d)$, then:

- i. $\text{atEntToVal}_d(i, x)$ is defined for all $x \in E_d^{AI}$,
- ii. $\text{atEntToVal}_d(i, x_1) \neq \text{atEntToVal}_d(i, x_2)$ for all distinct $x_1, x_2 \in E_d^{AI}$.

Within a web application domain, an identifier might be an email address which is dependent on a centralized email provider. However, in decentralized systems, we prefer cryptographically-generated identifiers such as DIDs, which guarantee global uniqueness without relying on any centralized provider. We can denote such an identifier (i) as $\text{DID} \in A^C$. For example, for a DeFi domain (denoted as d_{DeFi}) for AgentX , the identifier function could reveal something like this: $\text{identifier}(d_{\text{DeFi}}) = \text{DID}^{\text{AgentX}}$. It is to be noted that two distinct entities cannot share the same identifier value in the same domain. For example, AgentX 's $\text{DID}^X = \text{"did:ethr:AgentX"}$ differs from AgentY 's $\text{DID}^Y = \text{"did:ethr:0x742d..."}$, providing unambiguous reference to each entity.

Definition 5 (Identity Resolution). As identifier $i = \text{identifier}(d)$ is unique for every entity in the domain (Definition 4), we define the identity resolution function $\text{idValToEnt} : AV_d \rightarrow E_d \cup \{\perp\}$ as the inverse of the valuation function. For any value $v \in AV_d$:

$$\text{idValToEnt}(v) = x \iff \text{atEntToVal}_d(i, x) = v$$

If no such entity exists, it returns $\perp$.

Some attributes can identify AI agents but without uniqueness guarantees. These attributes act as *partial identifiers*: they assist in differentiating groups of agents, but they do not guarantee uniqueness within the entire domain. The unique identification of each agent instead relies on cryptographic attributes such as the DID in A^C . We define a partial identifier in the following way:

Definition 6 (Partial Identifiers).

$$PI_d = \{a \in A_d \setminus \{i\} \mid \exists v : |\{x \in E_d \mid \text{atEntToVal}_d(a, x) = v\}| \geq 1\} \quad (4)$$

By definition, $PI_d \subseteq A_d \setminus \{\text{identifier}(d)\}$ contains attributes whose values may identify one or more AI entities, but without uniqueness guarantees.

In the AI agent domain, several attributes naturally fall into this category. Let us assume that several agents may share the same `modelHash` if they are instantiated from the identical weights and architecture. For example, in domain d_{AI} , two

agents, AgentX and AgentY , are instantiated from the same release and therefore both share the same hash of model weights. Formally,

$$\begin{aligned} \text{atEntToVal}_d(\text{modelHash}, \text{AgentX}) &= 0x9af4 \\ \text{atEntToVal}_d(\text{modelHash}, \text{AgentY}) &= 0x9af4 \end{aligned}$$

This is an indication that although `modelHash` can cluster agents with the same release, it cannot alone differentiate between individual instances, and is therefore a partial and not a unique identifier. In a similar manner, two agents might share the same `versionTag` if they originate from the same release.

Identity Constructs: We now focus on the development of an agent's identity within a domain. This process occurs in three phases: *partial identity*, *profile*, and ultimately *total identity*, where each phase builds upon the previous one. Subsequently, We formalise what an identity is for an AI Agent within a domain, how profiles are constructed from its identity, and how identities across domains accumulate into a global view without losing provenance.

Definition 7 (Partial Identity). The **partial identity** of an agent $\text{AgentX} \in E_d^{AI}$ in domain d is the set of attributes defined for that agent in that domain:

$$\begin{aligned} \text{parIdent}_d^{\text{AgentX}} &= \{(a, v) \mid a \in A_d, \\ &\quad \text{atEntToVal}_d(a, \text{AgentX}) = v\}. \end{aligned} \quad (5)$$

That is, the partial identity is the set of attribute–value pairs (a, v) such that attribute a from the domain d attribute set A_d is defined for the AI entity x , and the valuation function atEntToVal_d returns a value v .

Consider our previous agent AgentX functioning within the DeFi domain, d_{DeFi} . In this context, important attributes consist of its decentralized identifier (DID), the hash of its model weights (`modelHash`), latency at the 95th percentile (`latency95`), and the decentralized identifier of its owner (`ownerDID`). The valuation function is defined for these attributes and yields:

$$\begin{aligned} \text{parIdent}_{d_{\text{DeFi}}}^{\text{AgentX}} &= \{(\text{DID}, \text{did:ethr:0x742d...}), \\ &\quad (\text{modelHash}, 0x3f2...), (\text{latency95}, 220\text{ms}), \\ &\quad (\text{ownerDID}, \text{did:ethr:Alice})\}. \end{aligned}$$

This set represents the **Partial Identity** of AgentX within d_{DeFi} . It contains only those attributes which the valuation function successfully defines for the agent in that domain.

While the concept of **Partial Identity** provides a comprehensive picture of everything the domain knows about a specific entity x in a certain domain, in practice agents rarely disclose their entire partial identity. Instead, they reveal a carefully selected subset of their attributes depending on the verification context. This leads to the notion of a **Profile**: a curated subset of the partial identity tailored for a specific interaction. This distinction is essential. Whereas, partial identity includes all the attributes specified by the domain that are available to

the entity, a profile is what the entity (or its rules) chooses to reveal in a particular context. This selective disclosure is particularly important in the case of AI agents in many situations such as regulatory audit, privacy-preserving checks, or delegating a contract, where not all information is needed or even can be dangerous. Consequently, every profile invariably includes the DID and may incorporate only the essential set of additional attributes required for a specific service or verification context. If A represents the specific attribute names the agent wishes to disclose, you must construct the attribute-value pairs dynamically in the union:

Definition 8 (Profile for AI Agents). For an agent $AgentX \in E_d^{AI}$, the profile of $AgentX$ in domain d is defined as:

$$PROFILE_d^{AgentX} = \{(identifier(d), v)\} \cup \{(a, atEntToVal_d(a, AgentX)) \mid a \in A\} \quad (6)$$

Here:

- $identifier(d) = DID$ is the unique identifier of the agent in the AI domain d ,
- $v = atEntToVal_d(DID, AgentX)$ is the specific identifier value of that agent,
- $A \subseteq PI_d$ is the chosen subset of partial identifiers the agent wishes to disclose, with the condition that $atEntToVal_d(a, AgentX)$ is defined for all $a \in A$.

Following the previous example for $AgentX$, let us assume that $AgentX$ interacts with a **compliance auditor** who only requires proof of the agent's existence and its linkage to its owner, but not its detailed performance metrics or architecture. In this scenario, the profile presented by the agent is:

$$profile_d^{AgentX} = \{(DID, did:ethr:0x72d...), (ownerDID, did:ethr:Alice)\}.$$

From this example, it is clear that a profile is a strict subset of the full partial identity of $AgentX$ in the DeFi domain. which remain part of the partial identity but are withheld for privacy and relevance reasons.

While the concepts of **partial identity** and **profile** describe what is known about an AI agent in a single domain, an agent rarely exists in isolation. Modern AI agents frequently operate across multiple domains simultaneously, for instance, an agent may function in a DeFi domain, while also being deployed in a supply chain ecosystem. In such cases, each domain maintains its own partial identity and associated profiles, but collectively, these must accumulate into a consistent global picture of the agent. This motivates the definition of **total identity**, a complete collection of all partial identities across all domains in which the agent participates. The total identity is defined as follows.

Definition 9 (Total Identity). For an agent $AgentX \in E_d^{AI}$,

the total identity is defined as:

$$identity^{AgentX} = \bigcup \{(d, parIdent_d^{AgentX}) \mid d \in D, AgentX \in E_d^{AI}\} \quad (7)$$

Here:

- $identity^{AgentX}$ denotes the global identity of agent $AgentX$,
- D is the set of all domains in which identities are managed,
- each component $(d, parIdent_d^{AgentX})$ pairs a domain tag d with the corresponding partial identity of $AgentX$ in that domain.

By formally unifying these fragments, the total identity preserves contexts while enabling reasoning about the agent across its entire operational scope. Let us return to the DeFi domain d_{DeFi} . Continuing with $AgentX$, we recall its partial identity in the **DeFi domain** $parIdent_{d_{DeFi}}^{AgentX}$.

Suppose the same $AgentX$ is also deployed in a **supply chain domain** d_{Supply} , where it monitors shipments and handles inventory transactions. Then its partial identity might be:

$$parIdent_{d_{Supply}}^{AgentX} = (DID, did:ethr:0x742d...), (maxShipmentValue, \$50,000), (auditLogHash, 0x8e12...).$$

The **total identity** of $AgentX$ is then obtained by uniting its partial identities across both domains:

$$identity^{AgentX} = \{(d_{DeFi}, parIdent_{d_{DeFi}}^{AgentX}), (d_{Supply}, parIdent_{d_{Supply}}^{AgentX})\}$$

This aggregated structure preserves the provenance of each fact (DeFi vs. supply chain), avoids attribute collisions, and provides a **global but domain-aware view** of the agent.

We must explicitly define how profiles function as subsets of the total identity. Profiles derive from their respective partial identities, and together, the partial identities contribute to the agent's **total identity**. Thus, profiles are the *contextual disclosures* of identity, while total identity is the *mathematical union* that secures coherence across all domains.

Cryptographic Binding: Cryptographic checks bind claimed identifiers to effective control. The following function abstracts signature verification.

Definition 10 (Cryptographic Proof). For an agent $AgentX \in E_d^{AI}$, the cryptographic proof of $AgentX$ in domain d is defined as:

$$checkProof_d^{AgentX} : ((DID_d^{AgentX}, v_i), \sigma) \mapsto \{\text{true}, \text{false}\}. \quad (8)$$

In decentralized systems, claims must be verifiable without trusting the claimant. When an AI trading agent claims to execute a transaction, it must provide a cryptographic signature proving control over its DID-bound private key. This signature can be verified by any party without requiring online interaction with the agent or its owner. For example,

If AgentX claims to have DID “did:ethr:0x742d...”, anyone must be able to verify that AgentX controls the private key corresponding to that DID. This prevents impersonation: merely knowing a DID is insufficient—you must prove control. A verifier accepts when the signature σ demonstrates control of the private key bound to the AI agent’s DID value v_i . This ensures that identity possession requires cryptographic proof of control rather than mere knowledge of the identifier. For example, when AgentX wants to prove its identity, it signs a challenge/message c as chosen by the verifier with its private key (K_X^{-1}): $\sigma = \text{Sign}(K_X^{-1}, c)$. Then $\text{checkProof}_d^{\text{AgentX}}((\text{DID}, \text{did:ethr:0x742d...}), \sigma) = \text{true}$ if and only if the signature verifies under the public key (K_X) registered in AgentX’s DID document. A successful verification proves that AgentX possesses the private key bound to the DID.

VI. THE ABCD FRAMEWORK

The ABCD framework organizes the identity of AI agents in four orthogonal dimensions, each can be verified independently, and these dimensions mutually reinforce each other. These layers are defined as restrictions or relations over various mathematical primitives, which reflect a multi-faced nature of AI agent identity, where computational, behavioural, cryptographic, and social aspects all need to be checked for complete trust establishment. We present each of these layers in the following.

A. Layer C: Cryptographic Identity

Before algorithmic or behavioural attributes matter, we need a cryptographically secure anchor. Layer C provides unforgeable digital identity through public-key cryptography and decentralized identifiers. The cryptographic layer serves as the cornerstone of AI agent identity, establishing the fundamental mechanisms for authentication, non-repudiation, and secure communication. In contrast to human identity systems, where cryptographic components may be secondary, for AI agents, cryptographic identity acts as the main trust anchor.

Definition 11 (Layer C: Cryptographic Identity). The DID attribute $\text{DID} \in \mathbf{A}_C$ is the domain identifier. For verification it uses the checkProof_d function.

This layer prevents impersonation attacks: without the private key, an attacker cannot forge signatures that pass checkProof_d . For AgentX with DID “did:ethr:0x742d...”, any action signed by AgentX’s private key creates an incorruptible proof. The cryptographic layer includes not only the DIDs but also associated public keys, key fingerprints for compact identification, and the verification procedures that bind cryptographic proofs to identity assertions, allowing them to engage in activities requiring non-repudiation proof.

B. Layer A: Algorithmic Fingerprint

Once cryptographic identity is established, Layer A defines a projection function Φ_{Alg} that extracts computational characteristics such as model, version, and configuration.

This function isolates the immutable computational properties (e.g., `modelHash`, `versionTag`) required for provenance verification. Function Φ_{Alg} is defined as follows.

Definition 12 (Algorithmic Fingerprint Function). We define the algorithmic fingerprint function $\Phi_{Alg} : E^{AI} \times D \rightarrow \mathcal{P}(A \times V)$ as a projection of the entity’s state of an agent AgentX :

$$\Phi_{Alg}(\text{AgentX}, d) = \{(a, v) \mid a \in A^A \wedge \text{atEntToVal}_d(a, \text{AgentX}) = v\} \quad (9)$$

The algorithmic fingerprint records the computational nature of an AI agent that is static, i.e. the model architecture, provenance of training, and configuration parameters. Typical components are elements of A^A and are static in nature. They change rarely and serve as provenance anchors, enabling long-term trust relationships based on verified computational properties. For AgentX:

$$\Phi_{Alg}(\text{AgentX}) = \left\{ \begin{array}{l} (\text{modelHash}, 0x3f2\dots), \\ (\text{architecture}, \\ \text{Transformer-6B}), \\ (\text{versionTag}, \text{v2.1}) \end{array} \right\}$$

If regulators require that only models approved by version v2.1 operate in a domain DeFi, they can verify $(\text{versionTag}, \text{v2.1}) \in \Phi_{Alg}(\text{AgentX})$. If AgentX were secretly running v3.0-beta, the version mismatch would reveal the violation. We note that impersonation resistance (Theorem 3) does not depend on Layer A. Moreover, in Section VII, Assumptions 2, 3, and 5 concern only key possession and identifier resolution, and Definition 6 already anticipates that `modelHash` may be shared, at negligible cost, across many agents from the same release. A cloned model therefore does not, by itself, grant a valid DID or private key for a different agent.

C. Layer B: Behavioural Fingerprint

Static provenance does not guarantee correct operations. Layer B captures empirical evidence: how the agent actually behaves over time. It captures the dynamic aspects of the operation of AI agents. The layer is important since AI agents, especially those that are machine learning-based, can also have emergent behaviour that cannot be entirely predictable based on static specifications.

Definition 13 (Behavioural Fingerprint Function). Let Φ_{Beh} be a function that returns the dynamic performance metrics of agent AgentX over time window W :

$$\Phi_{Beh}(\text{AgentX}, W) = \{(a, v) \mid a \in A^B, v = \text{atEntToVal}_d(a, \text{AgentX})\} \quad (10)$$

These capture how the agent behaves under load or policy (e.g., refusal rate, latency). As an example, for AgentX over the past

30 days ($W = 30d$), Φ_{Beh} could return something like the following:

$$\Phi_{Beh}(AgentX, 30d) = \{(\text{latency}95, 220\text{ms}), (\text{refusalRate}, 0.03), (\text{avgTradeSize}, \$5000)\}$$

In addition, let's say AgentX's documented $\Phi_{Alg}(AgentX)$ claims the model responds in under 200ms. But $\Phi_{Beh}(AgentX, 30d)$ shows 220ms latency. This shows incoherence flags because of potential issues such as model drift, resource contention, or malicious modification. As formalized in Corollary 2 (Section VII-C) and illustrated in VIII, the coverage of π scales with $|A_B|$ and with how tightly each τ_a is calibrated (e.g. via a $k\sigma$ control limit rather than a fixed constant). Latency and refusal-rate are illustrative instances, not an exhaustive set.

D. Layer D: Delegation and Ownership

Finally, the delegation and ownership layer addresses the fundamental question of accountability of the AI agent systems. The questions discussed in Layer D are questions concerning responsibility for the agent's actions and the authority under which it acts. Such a level is especially essential with the decentralized systems in which the hierarchies of the traditional organization may not apply.

Definition 14 (Delegation and Ownership). Let DID^u denote the DID of a human (user, $u \in E_d^u$), $Perm$ the set of permissions, and T the set of time intervals. Define:

$$\begin{aligned} \text{Own} &\subseteq DID_d^u \times E_d^{AI}, \\ \text{Del} &\subseteq DID_d^u \times E_d^{AI} \times Perm \times T \times T. \end{aligned} \quad (11)$$

Ownership binds an agent to the human who is responsible for it, whereas delegation gives the agent the right to act on a temporary basis $p \in Perm$. In order to implement **ST6**. (Elevation of Privilege) prevention, we practically implement these tuples based on the W3C Verifiable Credential (VC) Data Model [22]. Delegation is represented as a Delegation VC (acting as an unforgeable, agent-specific authorization token) with the DID of the human owner (DID^u) serving as the Issuer, and the DID of the agent (DID^{AgentX}) rigidly bound as the `credentialSubject`.

Let us assume for AgentX: $(\text{did:ethr:Alice}, AgentX) \in \text{Own}$. This establishes Alice as the accountable owner. Alice might delegate, allowing AgentX to trade up to \$50,000 between times t_1 and t_2 :

$$(\text{did:ethr:Alice}, AgentX, \text{trade_upto_50k}, t_1, t_2) \in \text{Del}$$

After t_2 , the delegation expires naturally. Nevertheless, in order to consider the case where early intervention is needed (e.g., the agent behaves maliciously prior to t_2), the framework supports explicit early revocation as described in Theorem 4.

E. Quadrant Relationships Between the ABCD Layers

Next, we argue that these layers are not merely a list of attributes but are structurally dependent on one another to form a complete identity. This is illustrated in Figure 1.

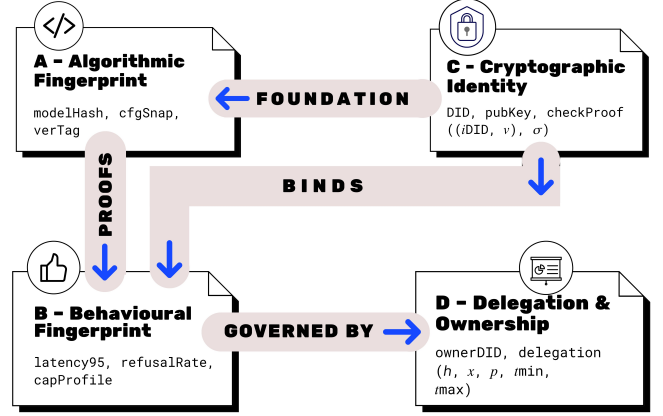


Fig. 1. Quadrant view of ABCD: Cryptographic foundation (C) binds algorithmic provenance (A) and behavioural evidence (B) under governance (D).

This quadrant relationship is elaborated in the following.

- **Foundation (C → A): The Cryptographic Identity (C)** serves as the immutable anchor. It binds the abstract digital identifier (DID) to the concrete **Algorithmic Fingerprint (A)**. Without C, the code in A has no verifiable owner; without A, the keys in C have no defined provenance. Concretely, key generation in Layer C (Definition 11) is cryptographically independent of Layer A. A DID/keypair can be minted with no algorithmic fingerprint at all. What Layer A's absence forfeits is not key validity but *provenance*. An unbound key can authenticate an entity but cannot prove which model that entity runs. Layer A's values are, in turn, intentionally not required to be unique across agents (Definition 6).
- **Binds (C → B):** The cryptographic layer also **Binds** the dynamic **behavioural Fingerprint (B)** to the entity. By signing runtime performance metrics and interaction logs with the private keys defined in C, the agent creates a tamper-evident link between its ephemeral actions and its identity.
- **Proof Generation (A → B):** The relationship between the static code and dynamic action is one of proof. The **Algorithmic Fingerprint (A)** establishes the baseline specifications (e.g., model architecture, latency targets) that serve as **Proofs** against which the runtime behavioural Fingerprint (B) is measured.
- **Governed By (B → D):** Finally, the agent's observable behaviour is governed By the **Delegation & Ownership (D)** layer. The specific actions recorded in B (e.g., executing a trade) are checked against the permissions and constraints defined in D to ensure they remain within the scope of human authorization.

F. Operational Architecture

Having defined the theoretical layers of the ABCD framework, we now envision the **Operational Architecture** required to implement them. As illustrated in Figure 2, the system is designed to support the full lifecycle of decentralized AI identity through three primary tiers: **Entities**, **Core Management Modules**, and **Decentralized Infrastructure**.

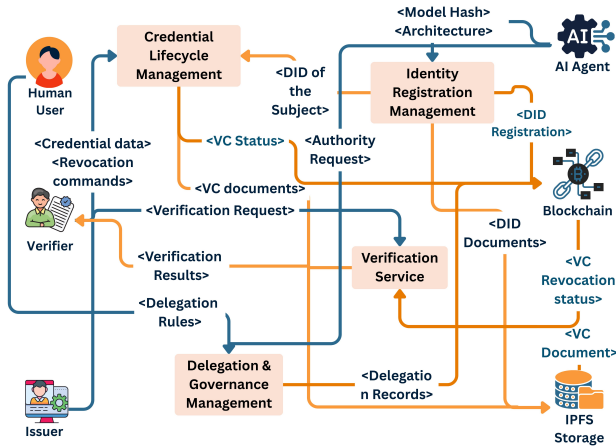


Fig. 2. Operational architecture of the identity framework

1) *Participating Entities*: The architecture supports four different actors that are defined in our domain model:

- **AI Agents:** These are autonomous entities which need to be registered and credentialed to engage in economic activity.
- **Human Users:** These are principals who possess AI agents and delegate authority to them. They communicate mostly with help of the Delegation and Governance Management module in an attempt to articulate scope and liability.
- **Issuers:** Trusted third parties (e.g., auditors, model vendors) which assert agent properties (e.g. model integrity or capabilities) through the Credential Lifecycle Management module.
- **Verifiers:** External applications (e.g., DeFi protocols) typically that query the Verification Service to verify the credentials of agents before granting them access.

2) *Core Management Modules*: The system logic is represented by four components:

- **Identity Registration Management (IRM):** The module manages the onboarding AI agents. It also acts as the gateway for both **Layer A** and **Layer C** where the raw model hash and architecture data are processed before the DID of the agent anchored to the blockchain.
- **Credential Lifecycle Management:** The issuance, update, and revocation of the VCs are done by this module. It guarantees that dynamic attributes (Layer B) and static attestations are signed cryptographically and stored correctly.

- **Delegation & Governance Management:** This module operationalizes Layer D, enabling human owners to establish time-bounded, permission scoped rules of delegation. It links the DID of the human to the DID of the agent and gives a definite line of liability.
- **Verification Service:** The module is the gatekeeper to the system. It combines the blockchain data with the decentralized data storage (i.e., IPFS) data to execute the multi-layered ABCD checks (coherence, impersonation, authorization) in real-time to the verifiers.

Decentralized Infrastructure: To guarantee sovereignty and privacy, the architecture employs a dual-storage strategy:

- **Blockchain:** Serves as the verifiable data registry (VDR), which offers an unalterable root of trust. It holds DIDs, public keys and cryptographic hashes (e.g. Model Hash, Policy Hash), however, never holds sensitive personal information or raw model weights.
- **Decentralized Storage:** Used for off-chain storage of heavy or sensitive data, such as full DID Documents, and detailed Delegation Records. Data here is content-addressed and encrypted, referenced on-chain only by its respective identifier.

G. Roadmap to Practical Implementation

Although the ABCD framework proposes a new identity calculus of AI entities, it does not involve the replacement of current internet protocols. Instead, it is intended to be deployed as an application layer on top of Self-Sovereign Identity (SSI) stacks and W3C standards that are mature and battle-tested. To illustrate how a system designer can transition out of this conceptualized model to a practical deployment, we describe the technology stack necessary below:

- **AI Identity Sidecar (ACA-Py):** Since most contemporary Agentic AI systems (e.g., LangChain [23], AutoGPT [24]) are written in Python, developers can implement the Core Management Modules (Figure 2) with Hyperledger Aries Cloud Agent Python (ACA-Py) [25]. ACA-Py is an open-source framework used to build SSI agent. It can expose REST APIs to facilitate SSI functionalities such as generating DIDx (Layer C), accepting Delegation VCs (Layer D) and interacting with other SSI agents to provide algorithmic and behavioral proofs (Layers A and B).
- **Bridging Human Authorization (OpenID4VC):** The framework uses OpenID to Verifiable Credentials (OpenID4VC) [26] to cope with the operational challenge of human-to-agent delegation. This is a protocol that connects the traditional Web2 identity infrastructure (e.g., OAuth 2.0) to decentralized identity. It enables a human user to authenticate with standard, existing authentication techniques (e.g., biometric authentication on a mobile device) and issue a mathematically constrained Delegation VC (Layer D) to the wallet of their agent, facilitated by ACA-Py. To be precise, about scope, all core verification in Layers A–D (Definitions 10–17) relies solely on standard digital signatures per Definition 10. ZKPs, discussed next,

are an optional, additive privacy layer, not a required primitive of the base framework.

- **Enterprise-Scale Infrastructure and ZKPs:** The underlying cryptographic primitives of this framework (W3C DIDs and VCs) are already being used to deploy large-scale systems. Enterprise-grade infrastructure, like the Microsoft Entra Verified ID [27] or the European Blockchain Services Infrastructure (EBSI) [28] are examples of such infrastructure that use these primitives and can be used to anchor Layer C by both systems designers and other infrastructure providers. Moreover, in order to counter the privacy-transparency trade-offs discovered in Layer B, developers may incorporate Zero-Knowledge Proof (ZKP) [29] stacks like Privado ID [30]. This enables the AI agent to cryptographically demonstrate compliance with a performance policy (π) to a DeFi verifier without the DeFi verifier seeing the underlying, proprietary operational data on which the proof was generated.

With the coordination of these already existing, open-source technologies, e.g., ACA-Py and OpenID4VC, and widely-deployed infrastructures, the ABCD framework can be implemented instantly to achieve decentralized agentic economies.

VII. SECURITY THEOREMS

To safeguard real-world use with reliable behaviour, we need policies that bind static definitions (algorithmic fingerprints, ownership, delegation) with dynamic checks (behavioural evidence, time windows, revocations). We first fix the coherence, impersonation, and authorization predicates as formal characterizations (VII-A), then make explicit the cryptographic and systems-level assumptions our guarantees rely on (VII-B), then state three theorems along with 2 corollaries capturing the corresponding security guarantees (VII-C).

A. Formal Characterizations

We record three predicates, i.e, coherence, impersonation, and authorization, continuing the Definition 1–14 sequence of V–VI. These are definitional: they fix what each predicate *means*. The theorems and corollaries (VII-C) then derive, under the assumptions of VII-B, the conditions under which each predicate is *guaranteed* to hold or fail. That derivation, not the definition itself, is the security claim.

A–B Coherence Failure: Formally, $Bound(AgentX, W)$ means there is coherence at window W for $AgentX$ and its negation ($\neg Bound(AgentX, W)$) states that a failure has been identified.

Definition 15 (A–B Coherence Failure).

$$\neg Bound(AgentX, W) \iff \exists \pi [\pi(\Phi_{Alg}(AgentX), \Phi_{Beh}(AgentX, W)) = false] \quad (12)$$

Here, $\pi : P(A^A \times AV_d) \times P(A^B \times AV_d) \rightarrow \{true, false\}$ is a coherence policy that measures whether an AI agent's (here suppose, $AgentX$) observed behavioural attributes $\Phi_{Beh}(AgentX, W)$ are consistent with its stated Algorithmic attributes $\Phi_{Alg}(AgentX, d)$ within a specified observation

window, W . If π returns false, the agent shows a **coherence failure**, which implies that runtime behaviour is not compliant with registered model configuration.

Impersonation: Let us define impersonation:

Definition 16 (Impersonation).

$$\begin{aligned} \text{Impersonates}(AgentX, v_i, \sigma) := & (\\ & checkProof_d^{AI}((DID, v_i), \sigma) = true) \\ & \wedge (idValToEnt^{AI}(v_i) \neq AgentX). \end{aligned} \quad (13)$$

Authorization: Let Revoked denote the set of revoked delegations, with $((u, AgentX, p, t_{start}, t_{end}), t_{rev}) \in \text{Revoked}$ recording that a delegation was revoked at time t_{rev} . Then

Definition 17 (Authorization).

$$\begin{aligned} \text{Authorized}(u, AgentX, p, t) := true \iff & ((u, AgentX) \\ & \in \text{Own}) \wedge \exists t_{start}, t_{end}. (u, AgentX, p, t_{start}, t_{end}) \in \text{Del} \\ & \wedge t_{start} \leq t \leq t_{end} \wedge \neg \text{Revoked}(u, AgentX, p, t). \end{aligned} \quad (14)$$

B. Assumptions

Assumption 1 (Collision Resistance). The hash function H (SHA-256) used to compute `modelHash` is collision-resistant: no probabilistic polynomial-time (PPT [31]) adversary can find $w \neq w'$ with $H(w) = H(w')$ except with negligible probability.

Assumption 2 (Signature Unforgeability). The digital signature scheme (Sign, Verify) underlying `checkProof` (Definition 10) is Existentially Unforgeable under adaptive Chosen Message Attack (EUF-CMA) [32].

Assumption 3 (Identifier Injectivity). The identifier function $DID \in A_C$ satisfies the injectivity condition of Definition 4; equivalently, `idValToEnt` is a well-defined, injective partial function.

Assumption 4 (Byzantine Fault Tolerant (BFT) [33] Determinism). The distributed ledger executing the smart-contract wallet logic maintains BFT consensus [34] with at most f faulty nodes out of $n \geq 3f + 1$, guaranteeing that all honest nodes deterministically agree on a single canonical execution history for every state transition.

Assumption 5 (Self-Certifying Identity Resolution).

Self-certification: Each entity's identifier value $v \in AV_{A_C}$ is deterministically derived from the entity's own public key at generation time (e.g. $v = f(K)$ for a collision-resistant f), consistent with the DID construction; consequently no adversary can pre-register or front-run another entity's future identifier value except by breaking Assumption 1.

Finality-gated reads: Any evaluation of `idValToEnt(v)` relied upon by a verifier is performed against the BFT-finalized state of the VDR (Assumption 4) at a confirmation depth beyond the network's reorganization bound, never against a mutable

local cache, an unconfirmed mempool entry, or an unverified resolver response.

Assumption 6 (Finality Semantics of Authorization Time). The time variable t used in $\text{Authorized}(u, \text{AgentX}, p, t)$ and $\text{Revoked}(u, \text{AgentX}, p, t)$ denotes the block-finalization time of the respective transaction on the canonical chain (per Assumption 4), not its network submission time or mempool arrival time. Ordering of two transactions that are both merely *pending* (submitted but not yet finalized) is not determined by this model.

Assumptions 1–3 are standard cryptographic hardness assumptions; Assumption 4 is the standard safety threshold for BFT ledgers (e.g., PBFT [34]) and is consistent with our system model (VI-F). Assumption 5 makes explicit two properties the paper already relies on informally (self-certifying DIDs, on-chain anchoring) so that Theorem 3’s guarantee cannot be defeated by attacking the *read path* to idValToEnt rather than the cryptography itself. Assumption 6 closes the gap between submission-time and finalized-time ordering that Theorem 4 otherwise leaves open to reordering attacks (Corollary 5). A further, model-specific hypothesis is stated locally within Theorem 1, since it concerns the *statistical* relationship between tampering and observed behaviour rather than a cryptographic hardness property.

C. Theorems and Proofs

Theorem 1 (Behavioural Coherence and Tampering Detection). *Let w^* be AgentX’s registered weights, with $\text{modelHash} = H(w^*)$ committed on-chain (Definition 12). Suppose an adversary Adv replaces AgentX’s active weights with $w' \neq w^*$ at runtime, and suppose further that this substitution is behaviourally sensitive: there exists a threshold τ and an attribute $a \in A_B$ such that $|\Phi_{Beh}(\text{AgentX}, W)[a] - \text{baseline}[a]| > \tau$ over the observation window W . Then the coherence policy π detects the tampering: $\neg \text{Bound}(\text{AgentX}, W)$.*

Proof: (Static divergence.) By Assumption 1, H is collision-resistant, so $H(w') \neq H(w^*)$ except with negligible probability; the tampered instance’s true weights no longer match the committed provenance, even though the on-chain record still reads $\text{modelHash} = H(w^*)$.

(Behavioural divergence.) Because AgentX’s observable behaviour over W is now generated by w' rather than w^* , the behavioural sensitivity hypothesis gives an attribute $a \in A_B$ with $|\Phi_{Beh}(\text{AgentX}, W)[a] - \text{baseline}[a]| > \tau$.

(Policy evaluation.) Take π to be the threshold coherence policy of VI-C, which returns **false** whenever any declared attribute in Φ_{Alg} is violated by more than its tolerance in Φ_{Beh} . By the previous step, $\pi(\Phi_{Alg}(\text{AgentX}), \Phi_{Beh}(\text{AgentX}, W)) = \text{false}$.

(Conclusion.) By the coherence characterization (Definition 15), the existence of one policy π with $\pi(\Phi_{Alg}, \Phi_{Beh}) = \text{false}$ is sufficient to conclude $\neg \text{Bound}(\text{AgentX}, W)$. Hence any behaviourally-sensitive tampering event is detected within W . ■

Remark 1. The behavioural-sensitivity hypothesis is not a cryptographic assumption but an empirical one, and it marks precisely where the guarantee’s boundary lies: an adversary whose tampering induces deviation below τ (a maximally stealthy model swap) falls outside its scope. This matches the limitation already discussed in X-A; Theorem 1 makes precise *when* detection is guaranteed, rather than claiming detection is absolute. We stress that the theorem is *conditional*, not circular: it assumes a stated, falsifiable structural property of the deployment and derives detection from it, exactly as a signature scheme’s unforgeability proof assumes a hardness property (e.g. Assumption 2) and derives a guarantee from it. What is genuinely open is *coverage*, for which real attacks does behavioural sensitivity hold; not the validity of the implication itself. Corollary 2 makes this coverage question precise. This also answers why detection proceeds via Φ_{Beh} rather than by re-checking $H(w')$ directly. Raw weights are deliberately never re-disclosed post-deployment (**ST4.**). So the on-chain modelHash cannot be re-verified live, only its downstream behavioural signature can.

Corollary 2 (Adversarial Evasion Characterization). *An adversary evades the guarantee of Theorem 1 against a tampering event $w^* \rightarrow w'$ if and only if, for every attribute $a \in A_B$ monitored by π and across the full input distribution AgentX encounters during W , the induced deviation stays within tolerance:*

$$|\Phi_{Beh}(\text{AgentX}, W)[a] - \text{baseline}[a]| \leq \tau_a$$

Proof: Immediate from the contrapositive of Theorem 1: π returns **false** (triggering $\neg \text{Bound}$) iff *some* monitored attribute exceeds its tolerance; hence π returns **true** (evasion succeeds) iff *no* monitored attribute does. ■

Remark 2. Corollary 2 reframes the detectability. Two levers shrink that region, both already latent in the framework:

- By increasing $|A_B|$, it enables monitoring more behavioural dimensions (e.g. task/data-type distribution, decision traces, not just latency) turns evasion into a higher-dimensional constraint problem; and
- By tightening τ_a via a principled statistical decision rule (e.g. a $k\sigma$ or CUSUM/EWMA [35] [36] control limit on each attribute’s baseline distribution) rather than an arbitrary constant.

Neither claims universal detection; both are concrete, falsifiable design knobs rather than an unexamined empirical premise.

Theorem 3 (Sybil and Impersonation Resistance). *Under Assumptions 2 and 3, for any registered agent AgentX and any PPT adversary $\text{Adv} \neq \text{AgentX}$ not possessing AgentX’s private key, the probability that Adv causes the system to accept it as AgentX i.e., produces (v_i, σ) with $\text{checkProof}_d^{AI}((\text{DID}, v_i), \sigma) = \text{true}$ and $\text{idValToEnt}^{AI}(v_i) = \text{AgentX}$, is negligible.*

Proof: Any such (v_i, σ) falls into exactly one of two cases.

Case 1 ($v_i = v_{\text{AgentX}}$, **AgentX’s own registered identifier**). By Definition 10 `checkProof` accepts only if σ verifies under the public key K_{AgentX} registered in AgentX’s DID document. If Adv produces such a σ without K_{AgentX}^{-1} , then Adv can be used to build a PPT forger $\mathcal{B}$ against K_{AgentX} (simulating AgentX’s public interactions as signing-oracle queries) that succeeds with the same probability. This directly breaks Assumption 2, so this case occurs with only negligible probability ε_{sig} .

Case 2 ($v_i \neq v_{\text{AgentX}}$, **some other registered identifier, e.g. Adv’s own** v_{Adv}). Here `checkProof` may accept trivially (Adv signs with its own key), but by Assumption 3, `idValToEnt` is injective, so `idValToEnt`(v_i) resolves to whichever entity registered v_i i.e. to Adv itself (or $\perp$), never to AgentX. This case therefore never satisfies the second conjunct of the acceptance condition, regardless of Adv’s computational power.

Since both cases fail except with the negligible probability ε_{sig} from Case 1, the overall success probability is negligible. ■

Remark 3. We use *Sybil resistance* here specifically for the impersonation sub-case (an adversary assuming a registered agent’s identity). Resistance to *mass creation* of new, individually-valid Sybil identities is a distinct property, provided not by this theorem but by the registration-cost argument of **ST5**. This tightens the case split, where Case 2 was argued to itself satisfy the impersonation predicate. Presenting one’s own validly-signed, correctly-resolving identifier is not an impersonation of AgentX. The predicate’s second conjunct (`idValToEnt`(v_i) $\neq$ AgentX in the general form of Definition 16) is *false* precisely in this case, which is why Case 2 poses no threat to AgentX’s identity. Only Case 1 is a live attack surface, and it is closed by EUF-CMA [32].

Theorem 4 (Temporal and Bounded Authorization Enforcement). *Under Assumption 4, for any agent AgentX, owner u , action p , and time t , the smart-contract wallet dispatches p on behalf of u via AgentX at time t if and only if*

$$\text{Authorized}(u, \text{AgentX}, p, t) = \text{true}.$$

Proof: By Definition 14 and Definition 17, `Authorized`(u , AgentX, p , t) is the conjunction of four predicates:

$$C_1 = [(u, \text{AgentX}) \in \text{Own}],$$

$$C_2 = [\exists t_{\text{start}}, t_{\text{end}}. (u, \text{AgentX}, p, t_{\text{start}}, t_{\text{end}}) \in \text{Del}], \quad C_3 = [t_{\text{start}} \leq t \leq t_{\text{end}}],$$

$$\text{and } C_4 = [\neg \text{Revoked}(u, \text{AgentX}, p, t)].$$

The wallet’s on-chain guard $G(u, \text{AgentX}, p, t)$ is smart-contract bytecode that reads the canonical registry state (via the Bitstring Status List lookup for C_4) and evaluates exactly $C_1 \wedge C_2 \wedge C_3 \wedge C_4$ before permitting dispatch of p .

By Assumption 4, all honest nodes agree on a single canonical state and execute G deterministically; no adversary (bounded by the BFT fault threshold f) can force G ’s committed output to diverge from $C_1 \wedge C_2 \wedge C_3 \wedge C_4$ evaluated against that canonical state. Consequently:

($\Leftarrow$) If `Authorized` = true, all C_i hold, so G evaluates true and — assuming an honest relayer submits the call (liveness) — the transition dispatching p is committed.

($\Rightarrow$) If p is dispatched via a canonical, committed transition, that transition only exists because G evaluated true on that state, which by construction requires $C_1 \wedge C_2 \wedge C_3 \wedge C_4$, i.e. `Authorized` = true. Conversely, if any C_i is false at t — e.g. $t > t_{\text{end}}$ violates C_3 , or a revocation recorded before t violates C_4 — deterministic evaluation forces G = false and the transition reverts.

This establishes the biconditional. ■

Remark 4. Unlike Theorems 1 and 3, this guarantee is not probabilistic conditional on a cryptographic hardness assumption; it follows from deterministic conjunctive evaluation under BFT consensus (Assumption 4). Its force lies in ruling out *any* execution path, including malicious relayers or reordering attempts within the fault threshold, that dispatches p while some C_i is false, not in the propositional logic alone.

Corollary 5 (Front-Running Mitigation via Challenge Window). *Suppose every delegated action p authorized under $(u, \text{mathsf{AgentX}}, p, t_{\text{start}}, t_{\text{end}}) \in \text{Del}$ above a materiality threshold is subject to a mandatory challenge window of Δ blocks between submission and execution eligibility, where Δ exceeds the network’s maximum expected finality latency λ with margin ($\Delta > \lambda$). Then any revocation submitted before an action’s challenge window opens is guaranteed, under Assumption 4, to be reflected in C_4 before that action becomes eligible for dispatch — closing the time-of-check-to-time-of-use race between a pending action and a racing revocation that Theorem 4 alone leaves open (Assumption 6).*

Proof: An action submitted at (finalized) time t_0 becomes eligible for dispatch only at $t = t_0 + \Delta$. A revocation submitted at any $t_{\text{rev}} \leq t_0 + \Delta$ is, by Assumption 4, finalized on the canonical chain no later than $t_{\text{rev}} + \lambda$, which by hypothesis satisfies $t_{\text{rev}} + \lambda < t_0 + \Delta = t$. Hence, by Assumption 6, C_4 is evaluated at dispatch-eligibility time t against a canonical state that already reflects the revocation, and Theorem 4 applies with `Authorized` = false. ■

Remark 5. This is the mechanism, not merely the assumption, that resolves the MEV/reordering exposure of Theorem 4: it is not that reordering cannot happen (a Dolev–Yao-class relayer can still reorder *pending* transactions freely), but that no reordering of two *pending* transactions can outrun a Δ -block mandatory delay once Δ exceeds finality latency with margin. We recommend the Operational Architecture (VI-F) adopt such a window for all Layer-D delegated actions above a materiality threshold, consistent with standard optimistic-execution patterns.

VIII. WORKED EXAMPLE: AGENTX AND ALICE UNDER A DOLEV–YAO ADVERSARY

We illustrate Theorems 1–4 and Corollaries 2 and 5 acting together, following AgentX, a DeFi trading bot owned by Alice. Throughout, the network attacker Adv is modelled as a standard

Dolev–Yao adversary: Adv has full control of the network – it can intercept, delay, reorder, inject, and replay messages, and can act as a relayer or block proposer within the Byzantine fault threshold f of Assumption 4 – but cannot invert one-way functions, forge signatures, or corrupt more than f consensus nodes. That is, Assumptions 1, 2, 3, 4, and 5 all hold *against* Adv; only their network position and message-scheduling power are unconstrained.

A. Setup (Layers C and A)

AgentX generates a keypair (K_X, K_X^{-1}) and derives its identifier $\text{DID}_X = \text{did:ethr:0x742d}\dots$ directly from K_X , satisfying self-certification (Assumption 5(i)). At block b_0 , AgentX anchors

$$\Phi_{Alg}(\text{AgentX}) = \{(\text{modelHash}, \text{sha256:4f7a2}\dots), \\ (\text{versionTag}, \text{v2.1}), (\text{latencyTarget}, 200\text{ms})\},$$

and Alice’s ownership $(\text{did:ethr:Alice}, \text{AgentX}) \in$ Own is recorded in the same block. Alice delegates trading rights:

$$(\text{did:ethr:Alice}, \text{AgentX}, \text{trade_upto_50k}, \\ b_0, b_0 + 216,000) \in \text{Del},$$

a 30-day window at a 12-second block time. Per Corollary 5, any single trade above a \$10,000 materiality threshold is subject to a mandatory challenge window of $\Delta = 64$ blocks (≈ 13 minutes) – chosen to exceed the chain’s observed finality latency of ≈ 20 blocks.

B. Day 15: Coherence Failure (Theorem 1, Corollary 2)

Over $W = 30\text{d}$, AgentX’s historical latency baseline has mean 200ms and standard deviation $\sigma = 15\text{ms}$; using a 3σ control limit (Remark following Corollary 2), $\tau_{\text{latency95}} = 45\text{ms}$. The platform observes

$$\Phi_{Beh}(\text{AgentX}, 30\text{d}) = \{(\text{latency95}, 280\text{ms}), \\ (\text{refusalRate}, 0.04)\}.$$

Since $|280 - 200| = 80\text{ms} > \tau_{\text{latency95}} = 45\text{ms}$, at least one monitored attribute exceeds tolerance; by Corollary 2 evasion has *not* occurred, so $\pi(\Phi_{Alg}, \Phi_{Beh}) = \text{false}$ and, by Theorem 1, $\neg \text{Bound}(\text{AgentX}, 30\text{d})$ is raised as an automated alert to Alice. (Had the deviation instead been, say, 30ms – within τ – this specific π would not flag it; that is the honest boundary of the guarantee stated in the Remark on Corollary 2, not a claim of universal detection.)

C. Same day: An Impersonation Attempt (Theorem 3)

While the alert is under review, a rival bot AgentY (controlled by Adv) attempts to hijack AgentX’s trading reputation with a counterparty.

- **Case 1.** AgentY presents DID_X with a signature it attempts to forge. Since Adv does not hold K_X^{-1} , producing a σ that verifies under K_X is an existential forgery; by Assumption 2 this succeeds with only negligible probability, and checkProof rejects.

- **Case 2.** AgentY instead presents its own validly-registered DID_{Adv} (self-certified from K_{Adv}) with a legitimately-signed σ_{Adv} , and separately *claims*, off-chain to the counterparty, to be AgentX. checkProof accepts trivially – but by Assumption 5, DID_{Adv} could not have been made to collide with or shadow DID_X without breaking Assumption 1, and the counterparty’s verifier resolves $\text{idValToEnt}(\text{DID}_{Adv})$ against the BFT-finalized registry, returning AgentY, never AgentX. The claim is rejected.

Both attack surfaces close: Case 1 by Assumption 2, Case 2 by Assumption 5.

D. Day 15–16: Revocation Under Contention (Theorem 4, Corollary 5)

Based on the coherence alert, Alice decides to revoke AgentX’s trading rights. Simultaneously, suppose AgentX – now suspected of running tampered weights – has an in-flight \$50,000 trade transaction sitting in the mempool, submitted at block b_1 with dispatch eligibility at $b_1 + \Delta$. Alice’s revocation is submitted at $b_1 + 5$ and finalizes at $b_1 + 5 + 20 = b_1 + 25$ (finality latency ≈ 20 blocks, Assumption 4). Because $b_1 + 25 < b_1 + \Delta = b_1 + 64$, the revocation is finalized *before* the trade becomes eligible for dispatch, regardless of how Adv – acting as a relayer or block proposer within the fault threshold – attempts to reorder the two pending transactions in the interim: Assumption 4 guarantees no execution path commits the trade against a canonical state that does not yet reflect the revocation. At $t = b_1 + \Delta$, the guard evaluates

$$C_4 = \neg \text{Revoked}(\text{Alice}, \\ \text{AgentX}, \text{trade_upto_50k}, t) = \text{false},$$

so G evaluates false and the transaction reverts – exactly as Corollary 5 predicts. (Had Alice instead revoked at $b_1 + 50$ – within 14 blocks of $b_1 + \Delta$ but after the trade’s own finality – the race would instead favor whichever transaction finalizes first; this is why Δ must exceed the network’s finality latency *with margin*, as Corollary 5 requires, not merely on average.)

E. Operational Note: Reconciling Finality with Machine-Speed Execution

Assumption 5(ii) requires finality-gated reads, but this does not require re-resolving identity and re-checking authorization on *every* micro-decision AgentX makes. Consistent with the Verification Service of VI-F2, a verifier resolves identity and authorization once per delegation window or per challenge-window boundary, caches the result for a short, explicitly bounded validity period, and invalidates the cache on a push event from the revocation registry rather than polling finality on every action. High-frequency actions inside an already-open, uncontested window proceed at cache speed; only delegation setup, revocation, and challenge-window-boundary checks pay the finality-latency cost – precisely the points at which Theorems 3 and 4 apply.

This walkthrough shows the four layers reinforcing one another operationally, as argued in VI-F: a Layer B anomaly (coherence failure) triggers a Layer D governance action

(revocation), whose correctness against a racing action depends on Layer C’s finalized resolution (Assumption 5) and Layer A’s cryptographic anchoring – turning the abstract quadrant relationship of Figure 1 into a concrete, adversarially-contested timeline.

IX. SECURITY ANALYSIS

In this section, we demonstrate how the ABCD Framework mitigates the specific threats identified in our Threat Model (Section IV-B).

Spoofing/Sybil Attacks (ST1.): We propose strict Entity Partitioning (E_d^{AI} vs E_d^U) and Cryptographic Proof of Control enforced by Layer C.

Analysis: Traditional systems bind acceptance to key possession alone via `checkProof`, with no independent provenance (Layer A) or behavioural (Layer B) check. A stolen, cloned, or freshly-generated key is therefore sufficient on its own to authenticate, regardless of how many other identities the same operator already controls. Our framework mitigates this by enforcing the Impersonation condition defined in Definition 16. The check $checkProof_d((i_{DID}^{AI}, v_i), \sigma)$ ensures that an attacker cannot simply replay a DID string. Furthermore, by rigorously partitioning the set of entities $E_d^u \cap E_d^{AI} = \emptyset$, the model prevents the identity laundering where human credentials are used to spawn unaccountable bot swarms.

Tampering/Model Swapping (ST2.): We propose a behavioural Coherence check that binds runtime metrics to Algorithmic Fingerprints ($\Phi_{Alg}(x)$) via Layer A/B.

Analysis: This addresses the black box vulnerability where a valid DID masks a malicious model. We enforce this with the security predicate $Bound_{AI}(x, W)$ (Definition 15). If an agent attempts to swap its underlying model for a cheaper or malicious variant, the resulting runtime behaviour $\Phi_{Beh}(x, W)$ (e.g., inference latency or decision variance) will statistically deviate from the signed $\Phi_{Alg}(x)$. This triggers a coherence failure ($\pi = false$).

Repudiation/Liability Evasion (ST3.): We propose non-repudiable links of delegation verified by Layer D.

Analysis: To prevent human owners from disowning the economic damage caused by their agents, we utilize the ownership relation $(did : ethr : Alice, AgentX) \in Own$. Because this link is cryptographically signed by the owner’s private key and anchored on the immutable VDR, it provides mathematical non-repudiation. An owner cannot retroactively deny the link after a catastrophic trade, as the chain of custody is preserved in the A^D attributes.

Information Disclosure (ST4.): We propose On-chain hash commitments combined with encrypted Off-chain storage (e.g. IPFS).

Analysis: We decouple verification from data visibility. The blockchain stores only the SHA-256 hashes of the model architecture ($modelHash \in A^A$), ensuring integrity. The sensitive weights and configuration files are stored off-chain.

Denial of Service (ST5.): We propose economic friction layers (e.g., gas fees) within Layer A.

Analysis: DoS attacks rely on the cost-free generation of identities. By requiring on-chain registration for Layer A (Algorithmic Fingerprinting), we impose a financial cost (gas) on the creation of every new agent identity. This economic barrier makes high-volume spam registration prohibitively expensive for attackers.

Elevation of Privilege (ST6.): We propose cryptographically enforced authorization constraints (A^D) via Layer D.

Analysis: In this model, privilege elevation manifests as exceeding delegated *scope* along any bound dimension such as permission p , time window $[t_{start}, t_{end}]$, or budget where encoded as a permission. These are all enforced identically by Definition 17’s conjunctive predicate and Theorem 4’s deterministic evaluation. Specifically, permissions are not static; they are strictly bounded by the tuple (p, t_{start}, t_{end}) . The system automatically rejects any action where $t > t_{end}$, ensuring that even if an agent is compromised, the attacker’s window of opportunity is strictly limited by the pre-defined delegation logic.

X. DISCUSSIONS & LIMITATIONS

The identity gap of decentralized Agentic systems is formalised by the ABCD Framework, with Algorithmic provenance, Behavioural evidence, Cryptographic anchoring as well as Delegated liability. Yet, the implementation of this calculus in permissionless infrastructure has certain trade-offs.

A. Statistical Nature of Verification

Unlike cryptographic signatures which provide binary verification (valid or invalid), Layer B relies on the Coherence Policy π to detect anomalies between declared Algorithmic Fingerprints (Φ_{Alg}) and observed behaviour (Φ_{Beh}). This introduces a fundamental limitation: verification is *probabilistic* rather than *deterministic*. The system acts as a statistical detector subject to false positives (e.g., benign performance drift due to network latency) and false negatives (e.g., adaptive adversaries minimizing divergence). The reliability of this trust model is therefore strictly dependent on the observation window W . A window that is too short increases noise and false alarms, while a window that is too long delays the detection of model swapping attacks.

B. Privacy-Transparency Trade-offs

While off-chain storage (e.g., IPFS) prevents the direct disclosure of model weights, a fundamental tension remains between public verification and intellectual property (IP) protection. To prove behavioural integrity (Φ_{Beh}), an agent must publicly disclose performance metrics (e.g., latency, refusal rates, decision traces). In competitive domains like DeFi, these behavioural traces can act as a side-channel. This can allow competitors to reverse-engineer proprietary strategies. For example, a consistent latency pattern might reveal the complexity of the underlying model. Future work must explore Zero-Knowledge Proofs (ZKPs) to allow agents to prove adherence to π without revealing the raw underlying metrics.

C. Attribution vs. Enforcement

Layer D successfully binds agents to human principals (E^U) cryptographically via the Own relation, solving the problem of *attribution* (ST3). However, it does not solve *enforcement*. If a pseudonymously owned agent causes financial damage, the protocol can prove who is liable, but it cannot compel restitution outside of pre-locked collateral. The framework provides the evidentiary basis for legal action but relies on external jurisdiction for execution.

XI. CONCLUSION

AI agents that act independently can disrupt established identity paradigms based on biological anchors. In this paper, we have presented the ABCD Framework, which is a domain-sensitive calculus that orthogonalises the conception of AI identity into Algorithmic provenance, Behavioral integrity, Cryptographic anchoring and Delegated liability. Here, human and AI entities were formally partitioned ($E^U \cap E^{AI} = \emptyset$) and three security theorems were formally proven under explicit cryptographic and systems assumptions. ABCD is not a definitive solution to the economic dilemma of autonomy, but a foundational standard. The opaque deployments are replaced with auditable guarantees, by binding what an agent is (Φ_{Alg}), how it acts (Φ_{Beh}) and who is accountable (E^U). Such a deterministic identity layer is a security requirement as Multi-Agent Systems become more and more the mediator of value.

REFERENCES

- [1] P. S. Viswanathan, "Agentic ai: A comprehensive framework for autonomous decision-making systems in artificial intelligence," *International Journal of Computer Engineering and Technology*, vol. 16, no. 1, pp. 862–879, 2025.
- [2] C. Wissuchek and P. Zschech, "Exploring agentic artificial intelligence systems: Towards a typological framework," 2025. [Online]. Available: <https://arxiv.org/abs/2508.00844>
- [3] M. Abou Ali, F. Dornaika, and J. Charafeddine, "Agentic AI: a comprehensive survey of architectures, applications, and future directions," *Artificial Intelligence Review*, vol. 59, no. 11, 2026. [Online]. Available: <https://doi.org/10.1007/s10462-025-11422-4>
- [4] J. Xie, Z. Chen, R. Zhang, X. Wan, and G. Li, "Large multimodal agents: A survey," 2024. [Online]. Available: <https://arxiv.org/abs/2402.15116>
- [5] T. South, S. Nagabhushanaradhya, A. Dissanayaka, S. Cecchetti, G. Fletcher, V. Lu, A. Pietropaolo, D. H. Saxe, J. Lombardo, A. M. Shivalingaiah, S. Bounev, A. Keisner, A. Kesselman, Z. Proser, G. Fahs, A. Bunyea, B. Moskowitz, A. Tulshibagwale, D. Greenwood, J. Pei, and A. Pentland, "Identity management for agentic ai: The new frontier of authorization, authentication, and security for an ai agent world," 2025. [Online]. Available: <https://arxiv.org/abs/2510.25819>
- [6] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic ai: Autonomous intelligence for complex goals—a comprehensive survey," *IEEE Access*, vol. 13, pp. 18 912–18 936, 2025.
- [7] D. Hardt, "The OAuth 2.0 Authorization Framework," Internet Engineering Task Force (IETF), RFC 6749, Oct. 2012. [Online]. Available: <https://www.rfc-editor.org/info/rfc6749>
- [8] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1," *The OpenID Foundation, specification*, vol. 335, 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html
- [9] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018.
- [10] M. S. Ferdous, F. Chowdhury, and M. O. Allassafi, "In Search of Self-Sovereign Identity Leveraging Blockchain Technology," *IEEE Access*, vol. 7, pp. 103 059–103 079, 2019. [Online]. Available: <https://ieeexplore.ieee.org/document/8777265>
- [11] R. Ranjan, S. Gupta, and S. N. Singh, "Loka protocol: A decentralized framework for trustworthy and ethical ai agent ecosystems," 2025. [Online]. Available: <https://arxiv.org/abs/2504.10915>
- [12] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, P. Bellavista, and M. Conti, "A survey on decentralized identifiers and verifiable credentials," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 6, pp. 3641–3671, 2025.
- [13] A. Dixit, M. Smith-Creasey, and M. Rajarajan, "A decentralized iiot identity framework based on self-sovereign identity using blockchain," in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, 2022, pp. 335–338.
- [14] O. Dib and K. Toumi, "Decentralized identity systems: Architecture, challenges, solutions and future directions," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 4, no. 5, pp. 19–40, Dec 2020. [Online]. Available: <https://ssrn.com/abstract=3785452>
- [15] M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele, and C. Allen, "Decentralized identifiers (dids) v1. 0," *W3C Recommendation*, vol. 19, 2022. [Online]. Available: <https://www.w3.org/TR/did-core/>
- [16] A. Fraser and S. Schneider, "On the role of blockchain for self-sovereign identity," in *Competitive Advantage in the Digital Economy (CADE 2022)*, vol. 2022, 2022, pp. 17–21.
- [17] K. Huang, V. S. Narajala, J. Yeoh, J. Ross, M. Lambe, R. Raskar, Y. Harkati, J. Huang, I. Habler, and C. Hughes, "A novel zero-trust identity framework for agentic ai: Decentralized authentication and fine-grained access control," in *2026 International Conference on AI x Data and Knowledge Engineering (AIDKE)*, 2026, pp. 98–101.
- [18] K. Huang, V. S. Narajala, I. Habler, and A. Sheriff, "Agent name service (ans): A universal directory for secure ai agent discovery and interoperability," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, 2026, pp. 1–9.
- [19] D. Feldman, E. Fox, E. Gilman, I. Haken, F. Kautz, U. Khan, M. Lam-brecht, B. Lum, A. M. Fayó, E. Nesterov *et al.*, "Solving the bottom turtle: a spiffe way to establish trust in your infrastructure via universal identity," *Sprint Lab, Nova Zelândia*, 2020.
- [20] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
- [21] M. S. Ferdous, G. Norman, and R. Poet, "Mathematical modelling of identity, identity management and other related topics," in *Proceedings of the 7th International Conference on Security of Information and Networks*, ser. SIN '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 9–16. [Online]. Available: <https://doi.org/10.1145/2659651.2659729>
- [22] M. Sporny, T. Thibodeau Jr., I. Herman, G. Cohen, and M. B. Jones, "Verifiable credentials data model v2.0," World Wide Web Consortium (W3C), W3C Recommendation, May 2025. [Online]. Available: <https://www.w3.org/TR/2025/REC-vc-data-model-2.0-20250515/>
- [23] V. Mavroudis, "Langchain," *Preprints.org*, 2024. [Online]. Available: <https://doi.org/10.20944/preprints202411.0566.v1>
- [24] M. Firat and S. Kuleli, "What if gpt4 became autonomous: The auto-gpt project and use cases," *Journal of Emerging Computer Technologies*, vol. 3, pp. 1–6, 08 2023.
- [25] OpenWallet Foundation, "Aries cloud agent python (aca-py) documentation and architecture," 2024, accessed: April 2026. [Online]. Available: <https://aca-py.org/>
- [26] OpenID Foundation, "Openid for verifiable credential issuance (oid4vci) specification," OpenID Foundation, Tech. Rep., 2024, accessed: April 2026. [Online]. Available: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
- [27] Microsoft Corporation, "Microsoft entra verified id whitepaper: Trustworthy, secure, and efficient digital interactions," Microsoft Corporation, Tech. Rep., 2024, accessed: April 2026. [Online]. Available: https://www.microsoft.com/content/dam/microsoft/final/en-us/microsoft-brand/documents/Microsoft-Entra-Verified-ID-Whitepaper_v5.pdf
- [28] European Commission, "European blockchain services infrastructure (ebsi): Verifiable credentials framework and trust registry," 2026, accessed: April 2026. [Online]. Available: <https://hub.ebsi.eu/vc-framework/>
- [29] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof-systems*. New York, NY, USA: Association for Computing Machinery, 2019, p. 203–225. [Online]. Available: <https://doi.org/10.1145/3335741.3335750>
- [30] Privado ID, "Privado id core architecture and zero-knowledge proof framework," 2024, accessed: April 2026. [Online]. Available: <https://docs.privado.id/docs/introduction/>

- [31] J. T. Gill, "Computational complexity of probabilistic turing machines," in *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, ser. STOC '74. New York, NY, USA: Association for Computing Machinery, 1974, p. 91–95. [Online]. Available: <https://doi.org/10.1145/800119.803889>
- [32] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 281–308, 1988. [Online]. Available: <https://doi.org/10.1137/0217017>
- [33] L. Lamport, R. Shostak, and M. Pease, *The Byzantine generals problem*. New York, NY, USA: Association for Computing Machinery, 2019, p. 203–226. [Online]. Available: <https://doi.org/10.1145/3335772.3335936>
- [34] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.
- [35] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954. [Online]. Available: <http://www.jstor.org/stable/2333009>
- [36] S. W. Roberts, "Control chart tests based on geometric moving averages," *Technometrics*, vol. 42, no. 1, pp. 97–101, 2000. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/00401706.2000.10485986>