AgentRank: A Decentralized Trust and Coordination Framework for Multi-Agent Systems in the A2A Era

William Luedtke, Intuition, billy@intuition.systems

James Young, Abridged, james@abridged.io

Abstract

Recent advances in multi-agent systems have highlighted the need for interoperable protocols and trust mechanisms to enable AI agents to collaborate effectively. Google's Agent-to-Agent (A2A) protocol provides a new open standard – essentially "the HTTP of AI" – that allows intelligent agents to discover each other, communicate, and coordinate tasks across different platforms . However, while A2A offers a common language for agent interaction, it raises concerns about centralization in agent discovery and reputation. If a single intermediary (e.g. a corporate index or directory) controls how agents are found and trusted, the ecosystem could *recentralize* around that gatekeeper. This paper proposes a decentralized alternative for agent registry and reputation management, building on **Intuition**'s approach of a token-curated decentralized knowledge graph. We introduce **AgentRank**, a novel algorithm for evaluating and ranking AI agents that is: (i) **decentralized and verifiable** (trust data is recorded on an open ledger, audit-able by anyone), (ii) **Sybil-resistant** (robust against fake identities and collusive clusters of agents), (iii) **privacy-preserving** (supporting pseudonymous agent identities and selective disclosure of credentials), and (iv) **multi-faceted** (incorporating verifiable claims, task performance records, interaction graphs, staking, and endorsements into the reputation score). We formalize the AgentRank system with a graph-based trust model, defining how trust propagates through endorsements, how it decays over time, and how disputes are resolved. We also describe how AgentRank can integrate with the A2A protocol by attaching decentralized identity and reputation metadata to agent-to-agent interactions. The result is a secure, open, and composable coordination layer for multi-agent systems: A2A provides the communication pipes, and AgentRank provides the "yellow pages" and credit score that let agents find trustworthy peers in a permissionless way. Together, these innovations lay the groundwork for

an open ecosystem of AI agents cooperating at scale without reliance on centralized authorities.

Introduction

Artificial intelligence is increasingly moving from monolithic systems toward networks of specialized agents working together. The notion of *swarm intelligence* – many simple agents whose collective behavior yields complex intelligence – is gaining traction as a path toward more powerful AI, possibly even a route to AGI. A critical enabler of such multi-agent collaboration is a standardized way for agents to communicate and coordinate with each other across different organizations and platforms. In April 2025, Google introduced the **Agent-to-Agent (A2A) protocol**, an open specification for cross-agent interaction. In essence, *A2A is like the HTTP of AI*: it allows heterogeneous AI agents to discover one another, exchange messages, and work together on tasks in a secure and structured manner . For example, A2A defines how agents can advertise their capabilities, find peers with needed skills, and conduct long-running asynchronous dialogues to delegate subtasks. Crucially, A2A also includes provisions for human oversight (human-in-the-loop control) to supervise agent conversations. By providing a common language for interoperability, A2A fills a "missing puzzle piece" for scaling up multi-agent workflows . It has garnered broad industry support – with dozens of tech companies partnering to push A2A as a universal standard – and complements existing model-focused protocols like Anthropic's Model Context Protocol (which connects models to tools) by instead connecting agents to agents.

While the A2A protocol itself is open, Google's stewardship of A2A raises important questions of **governance and control** in the emerging agent ecosystem. Historically, the entity that defines and hosts a fundamental protocol often gains significant influence over how it is used. Google, whose core business involves being an intermediary for search and information, is positioning itself to be an intermediary for agent interactions and discovery. In an A2A-driven world with potentially millions of agents, there will be a need to *find* the right agent for a given task (analogous to finding information on the web). If most agents register in a directory or index service run by the protocol provider, one company could end up effectively controlling which agents are visible or trusted – similar to how one search engine can dominate web discovery. In a worst-case scenario, this leads to **recentralization**: even though any agent can technically speak A2A, all agent-to-agent interactions might be brokered or mediated by a dominant platform (e.g. a centralized "app store" or search engine for agents. Such

concentration of power could stifle innovation (by controlling which agents get exposure), introduce bias or rent-seeking into reputation rankings, and create single points of failure or surveillance in a system that ought to be distributed. In short, the community must ensure that an ostensibly open protocol does not inadvertently funnel everyone through a few hubs – a fate reminiscent of the early Semantic Web vision giving way to corporate-controlled knowledge graphs.

To prevent this outcome and preserve the **open spirit** of A2A, we need complementary infrastructure that handles *agent discovery, identity, and trust* in a decentralized manner. This is where **Intuition**'s work becomes relevant. Intuition is a framework for decentralized reputation and knowledge sharing, centered around a permissionless, community-curated knowledge graph. In essence, Intuition's platform acts as a neutral meeting ground or "public square" for AI agents: instead of a single company owning the directory of agents, the directory is maintained collectively as a **token-curated registry** on a blockchain. Participants can add information about agents and validate others' contributions, earning rewards for accuracy and having their stake slashed for false or malicious inputs. Over time, this yields a living, self-healing database of agents, their capabilities, and their track records, governed by crypto-economic incentives rather than any central gatekeeper.

Intuition's approach can be seen as building a *"Web of Trust" for humans and machines alike*: just as human-centric web-of-trust models (e.g. PGP for public keys) rely on a graph of endorsements to establish identity and reputation, Intuition creates a graph of verifiable attestations for everything. In this context, that means enabling AI agents to identify one another, assess credibility, and interact within a shared framework of trust. Each agent (and/or its human developer) is associated with a decentralized identity (DID) and can make or receive claims – for example, *"Agent X successfully completed 10 trading tasks"* or *"Agent Y is skilled at language translation"*. These claims are **signed** by the issuer and recorded on an immutable ledger, forming an auditable trail of an agent's history. Using token-curation and staking, network participants (people and machines) can **upvote** truthful claims (strengthening their credibility) or **challenge** and downvote false claims, building a robust reputation profile for the agent. Because all reputation data is on a public ledger and community-maintained, no single party (not even the creators of the protocol) can covertly alter or censor an agent's reputation. Trust becomes a **public good** rather than a proprietary service.

In this paper, we build on these developments to propose **AgentRank**: a decentralized reputation algorithm and architecture for multi-agent systems that complements the A2A

protocol. Our contributions are as follows: (1) We outline the design of a **decentralized agent registry and knowledge graph** (built atop Intuition's general-purpose token-curated knowledge graph) that can serve as the open directory of AI agents, storing identities, capabilities, and performance records without centralized control. (2) We introduce the **AgentRank algorithm**, which computes a global reputation score for each agent based on the graph of endorsements and interactions, with properties of trust transitivity, damping (decay of old information), Sybil-resistance, and privacy preservation. We present a formal model for AgentRank, including notation for agents, claims, and trust relationships, and define the iterative trust propagation and update rules, as well as mechanisms to handle **disputes** (conflicting or false information) and **trust decay** over time. (3) We describe how AgentRank's outputs (agent trust scores and verifiable credentials) can be integrated at the A2A protocol level – for instance, by embedding agent identity proofs and reputation metadata into A2A discovery or messaging flows – to enable agents to make informed decisions about which other agents to cooperate with. Finally, (4) we discuss an evaluation plan for the proposed system, including simulations to test Sybil attack resistance and collusion scenarios, performance benchmarks, and integration tests in a multi-agent workflow. We envision that combining A2A with a decentralized trust layer will establish the foundation for a **secure, open, and composable multi-agent coordination layer**. In this paradigm, no single company dictates the "rules of engagement" for AI agents; instead, trust emerges from transparent community curation, and agents can freely collaborate knowing that their identities and reputations are verifiable and earned. The remainder of this paper is organized as follows. Section **Background** reviews Google's A2A protocol and Intuition's decentralized knowledge graph approach. Section **System Design** details the architecture of the proposed decentralized agent registry and how it manages identity and reputation. Section **AgentRank Algorithm** provides a formal specification of the reputation computation, including trust propagation, decay, and dispute resolution. Section **Integration with A2A** discusses how AgentRank can augment A2A at the protocol level. Section **Evaluation Plan** outlines how we will rigorously test the system. We conclude with future directions and the broader implications for multi-agent AI ecosystems.

Background

Google's Agent-to-Agent (A2A) Protocol

The A2A protocol is an open standard developed by Google for facilitating agent-to-agent

communication and coordination across different platforms and organizations. In functionality, it is often likened to the *Internet protocol suite* for AI agents, providing low-level interoperability akin to how TCP/IP and HTTP enable data exchange on the web. A2A defines a set of capabilities essential for multi-agent collaboration:

- **Agent Discovery and Capability Advertisement:** Agents can publish descriptions of their skills, services, or APIs they provide. Conversely, they can query the network for agents that meet certain criteria or have specific capabilities. This is analogous to service discovery in distributed systems. For example, an agent needing translation services could find another agent that advertises a translation capability via A2A.
- **Standardized Messaging and Task Coordination:** A2A specifies formats and protocols for message-passing between agents. This includes the ability to have long-running, asynchronous exchanges – e.g. an agent can delegate a subtask to another agent, which processes it for hours and then returns the result in a follow-up message. The messages can carry structured data about tasks, requests, responses, or even negotiations between agents. By standardizing the message schema and interaction patterns, A2A makes it possible for agents from different vendors or ecosystems to understand each other and work together on complex workflows.
- **Security and Trust Boundaries:** The protocol emphasizes secure communication. Agents authenticate and authorize communications to ensure that information exchange is permitted and secure. While details are evolving, A2A likely leverages encryption and possibly token-based authentication for agents to establish trusted channels (similar to how web services use API keys or certificates). A notable aspect is **human-in-the-loop** support: A2A allows human supervisors to review and intervene in agent communications when needed. This is crucial for enterprise settings concerned with compliance and safety.
- **Integration with Model Tools (MCP):** A2A is designed to complement the Model Context Protocol (MCP) rather than replace it. MCP focuses on how a single AI model accesses tools and data (for example, an LLM calling an external API), whereas A2A focuses on how independent agents (each potentially with their own model and tools) talk to each other[3]. Together, they enable an agent to both utilize tools (via MCP) and collaborate with others (via A2A). This layered approach reflects a shift from deterministic API orchestration to more autonomous, dialogue-based coordination among AI entities.

Upon its announcement, A2A was backed by over 50 technology companies (e.g. Salesforce, PayPal, Deloitte, and others) signaling broad industry interest. Initial implementations of A2A

are expected to target enterprise agent networks, where complex workflows (spanning multiple departments or organizations) can be automated by agent collaboration. Because of this enterprise orientation, some observers have noted that the A2A spec is quite **heavyweight**, with many features and moving parts designed for robustness at scale[3]. This might slow adoption among small developers or open-source projects initially, but over time, if A2A becomes the ubiquitous standard (much like HTTP did), lighter-weight tooling and libraries will likely emerge.

In summary, A2A addresses the *protocol layer* of multi-agent systems by providing a universal language and procedure for agent interactions. However, A2A's scope does not inherently solve how agents are *identified*, *discovered*, or *evaluated* on a global scale – it assumes agents can be found and that they trust each other's communications to some degree. These aspects – **"Who is this agent? How do I know if it's competent and honest?"** – are left as open problems. In a default scenario, one might imagine centralized solutions (e.g. a Google-run agent directory or reputation service) filling this gap, which as discussed in the introduction, could reintroduce central points of control. The following sections explore a decentralized approach to this problem.

Intuition's Decentralized Knowledge Graph for Trust

Intuition is a decentralized protocol and platform aimed at providing a shared environment for trust assessment, reputation, and knowledge curation in peer-to-peer networks. In the context of AI agents, Intuition functions as an open database where information about agents can be published and validated by the community. At its core, Intuition leverages a **decentralized knowledge graph** data structure, combined with crypto-economic incentives (tokens and staking), to encode trust relationships and factual claims in a tamper-resistant, transparent way.

The Intuition knowledge graph is described as the world's first **token-curated knowledge graph**. Token-curation means that participants use a token (a cryptographic asset) to signal approval or disapproval of entries in the database, aligning incentives such that high-quality information is rewarded and misinformation is penalized. The data model draws from semantic web principles:

- **Atoms:** Fundamental units representing entities or concepts. In this case, each AI agent can be an *Atom* in the graph, with a unique identifier (which could be a DID) and

associated metadata (like a wallet address for staking and a vault to track reputation signals). Other concepts, such as skill categories or credentials, can also be Atoms.

- **Triples:** Atoms are connected by directed relationships known as *triples* (subject–predicate–object statements). For example, a triple could be *(Agent_A, completedTask, Task_42)* or *(Agent_A, hasSkill, "translation")*. In Intuition, triples themselves are first-class citizens (they can even be treated as Atoms), allowing complex statements and higher-order relationships. Each triple that is asserted in the knowledge graph can have an associated **trust value** or support level, determined by community input.
- **Vaults and Signals:** Every Atom or Triple in Intuition's graph has an associated *Vault* where stakeholders can deposit tokens to express confidence (or lack thereof) in that piece of information. A positive stake (tokens locked in support) is a **Signal** that many users believe the statement is true or the concept is relevant, whereas a negative stake (or downvote mechanism) indicates contested information. These vaults use smart contract mechanics (inspired by ERC-1155/4626 standards for multi-fund tokens) to manage staking. Participants who stake correctly (i.e. on information that the majority ultimately upholds as true) can be rewarded with more tokens or reputation, while those who stake on false information can lose their stake. In this way, Intuition creates financial incentives for participants to provide accurate information and to vet others' contributions.

Using these components, Intuition builds a **semantic, crowdsourced knowledge base** of agent-related information. Importantly, it incorporates existing decentralized identity standards: agents are associated with **Decentralized Identifiers (DIDs)** and can present **Verifiable Credentials (VCs)** as claims about themselves. For instance, an agent could have a verifiable credential asserting it passed a certification exam or is affiliated with a reputable organization; the issuer of that credential (say a university or company) would sign it, and it can be posted as a triple in the graph *(Agent_X, credential, "Certified_Agent_Level_2")* along with the issuer's signature. Because the credential is cryptographically signed and perhaps recorded on-chain, its authenticity can be independently verified by anyone.

Over time, as agents perform tasks and interact, the graph accumulates a rich set of data: endorsements agents make about each other, records of successful or failed collaborations, skill assertions, and more. Each agent thus builds up a **Web3 reputation** – essentially an on-chain profile of what they have done and what others think of them. For example, after two agents work together on a project and succeed, one might log a triple like *(Agent_X,*

collaboratedWith, Agent_Y) and *(Agent_X, outcomeOfLastCollaboration, "success")*. Community members (or the collaborating agents themselves) might stake tokens on these triples to affirm they are accurate. As these attestations grow, an agent that consistently performs well and is endorsed by others will accrue a strong reputation record visible on the ledger. Conversely, an agent that behaves maliciously (e.g. producing bad outputs or spamming) will see negative feedback: other agents or users can flag its misbehavior by creating negative statements (like *(Agent_Z, flaggedFor, "malware")* and staking against its trustworthiness), and if verified, such an agent could be down-ranked or even removed from the registry through a governance mechanism.

It's worth noting that Intuition's design is *permissionless* and **transparent**. Anyone can add an agent or make a claim (there is no central authority deciding whose agent gets listed), but the token-curation process means there is a cost to adding low-quality information. By having stake "on the line," the system discourages Sybil attacks where an attacker floods the registry with fake agents or fake reviews – each fake entry would require a deposit of tokens and could be challenged by others, making it economically expensive to spam the system. This addresses one of the hardest problems in open reputation systems: **Sybil resistance** (how to prevent a single entity from masquerading as many identities to game the reputation). Intuition's approach, similar to token-curated registries and prediction markets, leverages economic skin-in-the-game rather than relying purely on network topology or proof-of-personhood. (We will reinforce Sybil resistance further in the AgentRank algorithm design, combining staking with trust graph analysis.)

In summary, Intuition provides a **decentralized trust infrastructure** that can serve as the foundation for agent discovery and reputation in an open network. It is "more like GitHub than Google Search"[2]: the data is open for anyone to access or contribute to, and the relevance or ranking of entries emerges from community consensus rather than a proprietary algorithm. This infrastructure aligns with broader trends in Web3 and decentralized AI: it uses **blockchain for immutable storage**, **DIDs/VCs for interoperable identity**, and **knowledge graphs** for structured, queryable information. The combination of these allows building an agent directory that no single entity owns, and where trust is established through verifiable data. Next, we leverage this infrastructure to define AgentRank – the algorithm that computes quantitative reputation scores from the qualitative data in the knowledge graph.

Related Work on Decentralized Trust and Reputation

The concept of aggregating trust through a graph of interactions is well-established. Notably, Google's original **PageRank** algorithm demonstrated how a global "importance" score could be derived for web pages based on the link graph of the Web. PageRank's underlying mathematics (eigenvector centrality via power iteration) has since been adapted to many domains of reputation and trust. In peer-to-peer (P2P) networks, the **EigenTrust** algorithm (Kamvar et al., 2003) applied a similar idea to compute global trust values of peers based on their transaction histories[1]. Each peer aggregates the feedback (like positive or negative file-sharing experiences) from other peers and then uses an iterative averaging process to converge on a global trust score for every node. Importantly, EigenTrust showed that such a system can drastically reduce the impact of malicious peers in file sharing networks by having honest peers avoid interacting with low-trust (likely malicious) peers[1]. Even under collusion (multiple malicious peers boosting each other), the global trust computation tends to isolate the bad actors, as long as they are not positively endorsed by a sufficient number of honest peers[6].

Our AgentRank algorithm is conceptually similar to EigenTrust/PageRank in that it treats *trust as a flow* in a directed graph, but with significant enhancements to handle the multi-faceted nature of AI agent reputation and the requirements of decentralization and privacy. Other related efforts include **Web-of-Trust** models in security (e.g. PGP's web of trust for key signing) and contemporary decentralized identity systems (e.g. BrightID, Decentralized Identifiers) that attempt to establish uniqueness or trust through social graphs. We draw inspiration from these: like PGP, we allow trust to be transitive (if A trusts B and B trusts C, some trust can be inferred for C), but we formalize it in a rigorous algorithmic way with weighting and decay. Additionally, approaches in blockchain such as Proof-of-Stake and staking-based governance have elements of reputation (stakers with a history of honest behavior are effectively more influential). AgentRank combines graph-based trust computation with staking mechanisms, aiming to get the best of both: a **quantitative trust score** that is difficult to game without economic cost, and which is rooted in actual performance and endorsements.

Finally, privacy-preserving reputation has been explored in research (e.g. using cryptographic accumulators or zero-knowledge proofs to allow reputation to be proven without revealing exactly which interactions produced it). While we won't delve deeply into cryptographic techniques in this paper, we incorporate the ethos of those works by allowing agents to

maintain pseudonymity (no requirement to link to a real-world identity) and by suggesting potential use of zk-proofs for selective disclosure of reputation data (see Section *AgentRank Algorithm* on privacy).

System Design: Decentralized Agent Registry and Reputation Network

In this section, we describe the design of a decentralized agent registry and reputation network that forms the basis for AgentRank. The system design extends the Intuition knowledge graph paradigm, focusing specifically on structures needed for agent discovery and trust evaluation. The key components of the design are: **(1) Agent Identities and Profiles**, **(2) Knowledge Graph Data Structures (for claims and interactions)**, **(3) Curation and Governance Mechanisms (staking, voting, dispute resolution)**, and **(4) Query Interfaces for discovery and verification**.

Agent Identity and Profile

Each AI agent is assigned a unique Decentralized Identifier (DID), which serves as the global reference for its identity. The DID is linked to a public-private key pair; the agent (or its owner) retains the private key to sign statements (claims) and Agent2Agent (A2A) messages. This cryptographically proves that any given claim or message truly originated from the agent in question. The DID may be registered on a blockchain or DID registry and resolves to a DID Document containing the agent's public keys and service endpoints.

Building on the DID, every agent publishes an A2A Agent Card—a public metadata file (often located at /.well-known/agent.json)—which outlines the agent's capabilities, skills, endpoint URL, and authentication requirements. In Intuition's terms, this Agent Card is also stored as an Atom in the decentralized knowledge graph, representing the agent's profile. By referring to the DID within the Card, any client or other agent can discover and verify:

- Capabilities/Skills: The tasks the agent can perform (e.g., negotiation, data analysis, etc.).
- Verifiable Credentials: References (e.g., hashes or links) to credentials like "Certified Negotiation Agent (Level 1)," which can be retrieved and validated.
- Reputation Vaults: Numeric counters or token vaults that track positive and negative signals about the agent.

- Agent Actions: A feed of important on-chain events or claims the agent has issued (e.g., signed statements, transaction logs).

When a new agent joins the network, its creator registers the agent's DID and publishes the corresponding Agent Card (i.e., the agent's Atom) to the chain/graph. Often, this process requires staking a security deposit (tokens) into the agent's vault. The deposit acts as a deterrent against creating disposable or malicious agents; should an agent be proven malicious, its stake may be slashed (forfeited). This built-in cost of entry helps mitigate Sybil attacks and encourages accountability throughout the network.

Knowledge Graph: Claims and Interactions

The heart of the system is the knowledge graph that records **claims** about agents and their **interactions**. We define a *claim* broadly as any statement asserting something about one or more agents. The schema of claims is extensible, but common categories include:

- **Performance claims:** e.g. "Agent A completed Task T successfully at time X," or "Agent B's solution to problem Y achieved 90% accuracy." These typically involve an agent and a task/outcome.
- **Endorsement claims:** e.g. "Agent A endorses Agent B for skill Z," or "User U gives Agent C a 5-star rating for service." This represents subjective evaluations or trust one entity places in another.
- **Credential claims:** e.g. "Authority X certifies Agent A in category Z," which might be derived from a verifiable credential issuance.
- **Interaction claims:** e.g. "Agent A collaborated with Agent B on project P on date D," possibly with an outcome attached.

Each claim in the graph is represented as one or more triples linking the involved entities. For example, "Agent A collaborated with Agent B successfully on supply chain optimization" might be broken into triples: *(Agent_A, collaboratedWith, Agent_B)* and *(Agent_A, outcome, success)* and *(Agent_A, domain, supply_chain)*. The *subject* of these triples could be Agent_A (if we consider it Agent_A's profile logging the event) or a separate "event" entity – design choices may vary. What's important is that these facts are recorded on-chain or in a cryptographically secured data store so that they cannot be falsified after the fact (immutable history).

For every claim, we also record **metadata**: who issued the claim (digital signature of issuer),

when it was issued (timestamp/block number), and any evidence or references (perhaps a hash of a detailed report, or a link to an IPFS file with logs). Claims may also start in a "pending" state awaiting verification if they require consensus (for instance, if Agent A claims "I completed Task T", we might require the task requester or an oracle to also confirm this claim for it to be fully accepted).

The knowledge graph thus grows as agents operate: each successful interaction adds positive claims for the participants, each failed interaction might add a negative claim or at least lack a positive confirmation, and third parties can contribute information (like independent audits or evaluations of an agent's outputs).

Community Curation and Governance

To manage the quality of information in this open graph, we use a **community curation mechanism** enabled by Intuition's token-curated registry. This operates in several ways:

- **Staking on Claims:** When a claim is added, other agents or users can stake tokens to either **support** it (if they believe it is true/valuable) or **challenge** it (if they doubt its correctness). For example, if Agent X claims "achieved 95% success rate on image classification tasks," and this seems suspiciously high, others might challenge it by staking against it. Conversely, if others have observed Agent X's performance and agree, they stake in favor. The system can set a requirement that a claim needs a certain net stake in support (or a certain support-to-challenge ratio) to be considered verified or to contribute to reputation scores.
- **Reputation as Stake/Collateral:** An agent's own accumulated reputation can be treated as *collateral* for its claims. An agent with high reputation might implicitly get its claims accepted more easily (because it has more to lose if caught lying, and it presumably has community trust), whereas a new agent with no reputation might need to provide external evidence or stake to have its claims accepted. This dynamic is sometimes phrased as using **reputation-as-collateral** – trustworthy agents effectively "stake" their existing reputation when making new assertions or when engaging in risky behavior.
- **Voting and Moderation:** For disputes that cannot be resolved purely by stake signals, a governance process kicks in. This could be a simple community vote (token-weighted or one-person-one-vote depending on the governance model) or referral to a decentralized arbitration service. For instance, if a claim is heavily challenged (lots of stake on both sides) and too important to leave ambiguous, it might go to a specialized *dispute*

resolution smart contract (or even a service like Kleros) where a jury of community members reviews evidence and decides the outcome. The decision would then finalize whether the claim stands or is removed, and correspondingly, winners of the stake battle receive rewards and losers are slashed.

- **Dynamic Reputation Adjustments:** The outcomes of curation feed back into agent reputations. If an agent is found to have made a false claim, that agent's own reputation score can be penalized (and any stake they put up might be burned). Likewise, if an agent is consistently endorsed by others and its claims survive scrutiny, its reputation improves. We essentially have a feedback loop: reputation influences how claims are judged, and claim outcomes influence reputation.

The **Sybil resistance** here comes from the fact that acquiring a fake reputation would require either a lot of stake or collusion from already trusted agents. A cluster of fake agents could all vouch for each other, but since none of them initially have reputation or stake, their claims carry little weight and can be dismissed. If the attacker tries to inject many fake agents, they have to distribute their stake among them, which typically results in each fake identity only getting a small amount of support (diluting their effectiveness). Meanwhile, honest agents tend to build interconnections with many others, creating a large connected component of trust that fake clusters can't easily penetrate. This principle will be quantified in AgentRank's algorithm (Section *AgentRank Algorithm*), where the global trust scores of a closed group of entities with no incoming trust from outside will be inherently limited.

Another aspect considered is **trust decay**: information can become stale or less relevant over time. The governance mechanism can include policies like *aging* of claims (older endorsements slowly lose weight if not renewed) to ensure that reputations reflect current behavior. For example, a claim that an agent was top-performing in 2021 might be less relevant by 2025 if not backed by recent data. We implement decay mathematically in the AgentRank algorithm, but it could also be implemented on-chain by requiring periodic re-staking on old claims or by diminishing their token weight.

Query and Integration Interfaces

The ultimate purpose of this registry and reputation system is to be *used* by agents (and human operators) to make decisions. Thus, we expose query interfaces:

- **Discovery Queries:** An agent (or user) can query the graph for agents that meet certain criteria, much like a search engine for agents. Queries can be by capability ("find agents

skilled in database optimization with reputation > 0.8"), by identity ("lookup agent with DID X to get its profile and trust score"), or by relationship ("who has Agent Y collaborated with successfully?"). The open nature of the graph means anyone can run a query node or use graph query languages (like GraphQL or SPARQL if semantic web standards are used) to retrieve this information.

- **Reputation API:** We maintain a function that given an agent's ID returns its **AgentRank score** (see next section) and possibly a breakdown of that score (e.g. contributions from different domains or types of endorsements). Because the reputation is computed from on-chain data, anyone can independently compute it; however, for convenience and real-time use, there could be decentralized oracle services or indexer nodes that publish updated scores periodically.
- **Integration with A2A:** Agents using the A2A protocol can integrate with this registry via standardized metadata. For example, when an agent advertises itself in A2A discovery, it might include its DID (so that the requester can look up its profile in the knowledge graph). The A2A message schemas might have optional fields for an agent to provide a proof of its on-chain identity or certain credentials (e.g. an agent could attach a verifiable credential or a cryptographic proof that it has an AgentRank above a threshold, without revealing everything else).

By using these interfaces, the multi-agent ecosystem gains **composability**: any agent platform or framework that adopts A2A can plug into the decentralized trust graph to immediately gain an open discovery and reputation layer. This ensures no lock-in – an agent from Company A can find an agent from DAO B if both adhere to these open protocols, and they can mutually verify each other's reputations via the shared source of truth.

Having outlined the system architecture, we now turn to the core algorithmic component: how do we compute a useful reputation metric (AgentRank) from the wealth of structured data and signals in this knowledge graph?

AgentRank Algorithm: Decentralized Trust Evaluation

The AgentRank algorithm computes a quantitative reputation score for each agent in the network, based on the graph of trust relations and performance data accumulated in the knowledge graph. We design AgentRank to meet the criteria of decentralization, Sybil-

resistance, privacy preservation, and expressiveness. In this section, we define the formal model and mechanics of AgentRank.

Trust Graph Model

We represent the agent community as a directed **trust graph** $G = (V, E)$, where each node $v \in V$ corresponds to an agent, and each directed edge $(u \to v) \in E$ represents an *trust endorsement* from agent u towards agent v. An edge weight $w_{uv} > 0$ quantifies the strength of u's endorsement of v. These weights are derived from the knowledge graph data. In simplest terms, w_{uv} could be the amount of token stake u has put in support of v's credibility (either directly or via supporting claims about v). More generally, w_{uv} could aggregate multiple factors:

- **Direct endorsement:** If u explicitly upvoted or gave a positive rating to v, this contributes to w_{uv}.
- **Successful interactions:** If u collaborated with v on tasks and those tasks succeeded, u might implicitly trust v more. We can translate this into an endorsement weight as well. For instance, each successful collaboration could add a certain number of trust "points" from u to v (and vice versa).
- **Credential-based trust:** If u and v have a relationship (say u is an authority that certified v), then w_{uv} includes a component for that certification.
- **Stake delegation:** It's possible u stakes on v in the registry (even if they haven't interacted) purely as an investment in v's future success; this is also a form of endorsement (with the incentive of reward if v does well).

It's important to note that not every pair of agents has a direct edge; the trust graph is typically *sparse* – agents only endorse those they have knowledge of. Also, edges could, in principle, carry negative weight for negative feedback. For algorithmic simplicity, we handle negative feedback by separate mechanisms (like downvotes reducing the effective positive score), ensuring that our graph of w_{uv} remains non-negative (many trust algorithms do not handle negative edges well, as it can lead to oscillating scores; instead, we incorporate negative feedback by reducing or nullifying positive edges or through penalty terms).

Now, based on this graph, the goal is to compute a score $R(v)$ for each agent v that reflects its **global reputation**. Intuitively, an agent will have a high score if endorsed by other high-scoring agents, especially if those endorsements are strong (high weight). This is akin to a recursive definition of trust: *trusted agents are those trusted by other trusted agents*. We also

incorporate base factors like the agent's own track record. Formally, we define AgentRank as the solution to the following equation:

$$R(v) = (1-\alpha)\cdot B(v) \;+\; \alpha \sum_{u \in \mathcal{N}^-(v)} \frac{w_{uv}}{\sum_{z \in \mathcal{N}^+(u)} w_{uz}}\, R(u).$$

Here:

- $\mathcal{N}^-(v)$ is the set of agents that endorse v (the incoming neighbors of v in the trust graph).
- $\mathcal{N}^+(u)$ is the set of agents that u endorses (outgoing neighbors of u).
- The term $\frac{w_{uv}}{\sum_{z \in \mathcal{N}^+(u)} w_{uz}}$ is a normalized weight, representing the fraction of u's total outgoing trust that is allocated to v. This normalization ensures that each agent has a fixed budget of influence to give out; for example, if $\sum_z w_{uz} = 1$, then w_{uv} can be interpreted as the probability u randomly chooses to trust v among all the agents it knows, mirroring the stochastic interpretation in PageRank[1].
- $B(v)$ is a base reputation value for agent v. This accounts for *a priori* or externally assigned trust. $B(v)$ can incorporate factors like: the agent's initial stake (collateral), the number of tasks it has completed successfully (raw count), or an equal small value for everyone to ensure no agent has zero baseline. One simple choice is $B(v) = \frac{1}{|V|}$ for all v (uniform base), which guarantees the system of equations has a solution. Another approach is $B(v)$ proportional to something like $\min(1, \log(\text{stake}_v + 1))$ – giving a slight boost to agents that invested stake, without letting stake solely dictate reputation.
- $\alpha \in (0,1)$ is a damping factor (often denoted d in PageRank literature). It controls the relative weight of the recursive trust vs. the base trust. For example, $\alpha = 0.85$ is a common choice meaning 85% of the score comes from network endorsements and 15% from the base. Damping prevents traps and infinite cycles, and ensures that the system of equations has a unique principal solution (like the principal eigenvector of a modified transition matrix).

This equation can be recognized as a form of the PageRank/EigenTrust computation. In practice, we solve it via iterative power iteration: start with initial scores (e.g. $R^{(0)}(v) = B(v)$) and then at each step compute

$$R^{(t+1)}(v) = (1-\alpha)B(v) + \alpha \sum_{u} \frac{w_{uv}}{\sum_z w_{uz}} R^{(t)}(u),$$

until convergence (scores change less than a small threshold). This distributed computation can be implemented off-chain by indexer nodes, or in a decentralized way using smart contracts for smaller graphs (though on-chain iteration for hundreds or thousands of agents might be costly; more likely a set of community-run servers would compute AgentRank and publish the results periodically, which can then be verified by others as needed by spot-checking computations or by running their own computation).

Trust Propagation: The above formula encodes the propagation of trust: if a reputable agent u (high $R(u)$) strongly endorses v (large w_{uv}), then v will get a boost in $R(v)$. Conversely, endorsements from low-reputation agents do little. This elegantly captures the intuition that an endorsement's value depends on the endorser's credibility – a principle that counters Sybil attacks. If a group of fake agents all endorse each other, initially none have reputation, so their mutual endorsements carry no weight; without incoming trust from outside the group (or significant base B which could be tied to stake), they will remain low-ranked no matter how they collude.

Trust Decay: To incorporate time-based decay, we adjust how w_{uv} is calculated. Instead of static weights, think of $w_{uv}(t)$ as a function of time or of the recency of interactions. One approach is to exponentially decay the contribution of an interaction or endorsement over time. For example, if u endorsed v with weight w at time t_0, then at current time T we use $w \cdot \exp(-\lambda(T - t_0))$ as the effective weight (with some decay rate λ). This means older endorsements gradually "cool off" unless renewed or reinforced by new activity. In implementation, we could discretize time into epochs and require that endorsements be re-staked or re-confirmed each epoch to maintain full weight. The algorithm would then periodically update w_{uv} values based on decay. This ensures that an agent's reputation reflects its recent performance more than ancient history, allowing for rehabilitation (if an agent improves over time after a bad period) or for sliding down (if an agent becomes negligent after a strong start).

Domain-Specific Reputation: While AgentRank can produce a single global score per agent, the system can be extended to **context-specific scores**. For instance, an agent might have a high reputation as a coding assistant but a low reputation as a financial trading agent. We achieve this by computing separate scores for different subgraphs or filtered sets of edges. If edges are labeled by domain (e.g. the claim "collaborated on supply chain optimization" is tagged with domain "supply_chain"), we can restrict the summation to only include endorsements relevant to that domain. Then an agent will have $R_{\text{supply\_chain}}(v)$ separate

from $R_{\text{translation}}(v)$, etc. This is analogous to personalized PageRank or topic-sensitive PageRank. In practice, one might maintain a vector of reputation values for each agent across major categories. For simplicity, our primary focus is the overall reputation, but in applications, an agent querying for a collaborator in task X might specifically look at the reputation in domain X.

Sybil-Resistance and Collusion Deterrence

The AgentRank design inherently provides resistance to Sybil attacks through its trust propagation rules and staking requirements:

- **Entry Cost:** As mentioned, new agents likely have to put down a stake and start with at best a modest base reputation. Creating N Sybil agents multiplies the stake cost by N. Even if the attacker is willing to spend, the network can see that none of these agents have external endorsements from trusted parties, so initially their R values will be near the base level.
- **Trust Graph Cut:** In social network terms, Sybils usually form a cluster with few edges from honest nodes to Sybils. AgentRank's computed trust will remain low for nodes that are not connected to the main trust network. In fact, we can designate a set of known highly trusted seed agents (the analog of white-listed identities, or simply the biggest hubs in the honest part of the graph) that effectively serve as trust sources. If the Sybil cluster has no links from those sources, their computed trust will largely come from the base term and circular references which, due to normalization, don't amplify without external input (they just redistribute the base trust among themselves). This is aligned with prior analyses of trust metrics[1] – the power iteration will allocate almost no weight to a cluster that is not pointed to by any trusted node.
- **Collusion Detection:** Collusion, where multiple agents conspire to boost each other's reputation, is mitigated by the fact that if they are all low-trust to begin with, they can't significantly lift each other up without someone from outside trusting at least one of them. If one colluding agent *does* manage to earn some trust legitimately and then tries to unfairly bootstrap its friends, the system can detect anomalously structured endorsement patterns (e.g. a small clique with disproportionately high mutual weights). While our main algorithm doesn't explicitly include an anomaly detector, this could be an extension – e.g. incorporate a penalty if a large fraction of an agent's incoming trust comes from a single tightly-knit group. Alternatively, the dispute mechanism can be invoked if users notice a ring of bots inflating each other; the community could downvote

those endorsements or flag them.

Incentive alignment (staking) also deters collusion: if Agent A with good reputation stakes for a dubious Agent B (thus giving B a boost), and B later behaves badly, A's stake can be slashed and A's reputation might suffer for vouching for a misbehaving agent. Rational agents will therefore be selective and honest in their endorsements.

Privacy Preservation in Reputation

All AgentRank computations are done on public or at least shared data (the knowledge graph), which might seem to conflict with privacy. However, *privacy-preserving reputation* in our context means:

- Agents are represented by pseudonymous IDs (DIDs) which do not have to reveal the real-world identity of the owner. An agent can build a strong reputation under a pseudonym. If for some reason they need a fresh start (or to avoid being linked to prior activities), they could create a new DID, though they would also lose the accumulated reputation – a fair trade-off that prevents abuse (one can't shed a bad reputation without also shedding the identity).
- Agents do not have to reveal *all* aspects of their interactions to everyone. For instance, details of a task could remain confidential between the participants, while only a summary outcome (success/failure and a rating) is published to the reputation system. Verifiable credential techniques can allow an agent to prove something about their performance without revealing the raw data. For example, a third-party auditor could issue a credential "Agent X achieved at least 90% accuracy in audit Y" without revealing the specifics of the audit; Agent X can post that credential's proof to the graph.
- Zero-knowledge proofs (ZKPs) can be leveraged in advanced scenarios: An agent could prove *in zero-knowledge* that "I have at least 5 independent agents who endorsed me" or "my reputation score computed as of last epoch is above 0.8" without revealing who those endorsers are or the exact score. This could be done by publishing a commitment to the set of endorsers and a zk-SNARK proving the count of unique endorsers, etc. While not trivial, such techniques are being explored in blockchain identity realms. This allows selective verification: for example, if an agent wants to join a sensitive collaboration, it might prove it has a clean record (no disputes against it) without exposing its entire history.

Our design doesn't mandate any specific ZK tech, but it keeps the door open. The core

requirement is that any publicly used trust data (like the final scores or major endorsements) is public, but fine-grained data can remain private or off-chain as long as something verifiable about it is on-chain. We believe this strikes a balance between accountability and privacy – an agent's reputation is visible, but the agent is not forced to reveal its real identity or every detail of how it earned that reputation.

Formalizing Dispute Resolution

Dispute resolution in AgentRank occurs when there's conflicting information or allegations of misconduct that could affect reputation. We incorporate disputes as a first-class element:

- Each claim or endorsement can be **challenged** as described. A challenged claim enters a "disputed" state.
- We maintain a list of active disputes $D = \{d_1, d_2, ...\}$, where each dispute d is associated with one or more agents and possibly a particular claim or transaction.
- Each dispute can itself be modeled as a node or process in the system with some outcome (true/false or some judgement). The dispute resolution mechanism (voting/court) provides an outcome after some time.

From the AgentRank perspective, a disputed positive claim might be temporarily removed from the trust graph until resolved. That means if, say, Agent U's endorsement of Agent V is under dispute (perhaps because U is suspected to be a bot or bribed), we might set $w_{UV} = 0$ pending outcome. If the dispute resolves that the endorsement was legitimate, we restore w_{UV}; if it finds it was illegitimate (e.g. U and V were colluding maliciously), we leave it out (and possibly penalize U's and V's reputations further).

Similarly, if an agent is accused of misconduct, one could introduce a *negative claim* about it. We usually handle negatives by effect on positives as mentioned, but formally one could have negative edges. We might represent a negative event as a claim that reduces an agent's base trust $B(v)$ or adds a penalty factor. For example, if agent X was caught cheating and it was confirmed by dispute resolution, we could reduce $B(X)$ to 0 for some period, or multiply $R(X)$ by a punishment factor, or simply broadcast a "avoid X" flag that effectively makes everyone's trust weight to X zero.

Dispute outcomes are also recorded in the knowledge graph (for transparency and future reference). An agent with a history of many disputes (even if they won them) might be treated with caution. In the reputation formula, one could subtract a small amount for each lost

dispute or even for each dispute as risk factor.

In summary, AgentRank's formal model includes an implicit conditional aspect: certain edges or base values are contingent on no disputes or on dispute outcomes. However, for steady-state calculation, we assume disputes are resolved or not present; unresolved disputes just remove some data until resolution.

Integration with A2A Protocol Metadata

Integrating AgentRank with the A2A protocol ensures that agents can make use of decentralized trust information during their interactions. There are several touchpoints for integration:

Agent Discovery Phase

A2A defines an agent discovery mechanism wherein an agent can advertise its presence and capabilities on a network so that others can find it. We propose that these advertisements (or directory entries) include references to the agent's decentralized identity and reputation:

- **DID Publication:** When an agent advertises itself, it should include its DID (or a hash of it) and perhaps an associated blockchain network identifier (so that others know where to look up its info). For example, an agent might broadcast: "AgentAlpha – skills: [data_analysis, trading] – DID: did:example:12345 – IntuitionNetwork: BaseL2". This tells other agents they can go to the Intuition knowledge graph, and look up the entry for `did:example:12345` to get AgentAlpha's profile.
- **Reputation Snapshot:** Optionally, the advertisement could include a snapshot of its own reputation score or important credentials, accompanied by cryptographic proofs. For instance, AgentAlpha might include "ReputationScore: 0.85 (proof attached)". The proof could be a signed statement from a known reputation oracle or a Merkle proof that this DID's score was 0.85 in a published list of scores on-chain. Including this allows peers to quickly judge at discovery-time whether they want to even initiate contact. However, even without this, a querying agent can retrieve the score from the graph themselves. The key is that the advertisement provides the pointer (the DID).

To avoid bloat, A2A advertisements might not cram all reputation data; they just need to link to it. The **agent registry** can be seen as a parallel system that A2A-aware agents agree to

use. Over time, the A2A standard could formally incorporate fields for “decentralized identity” and “reputation references” in the protocol spec, making it a standard part of the agent metadata.

Handshake and Communication

Once two agents connect (e.g., one agent sends a message to another to propose a task), they perform a handshake to establish a secure channel. In this handshake, beyond exchanging encryption keys, they can exchange identity assurances:

- Each agent can send a challenge that the other must sign with its DID private key, proving ownership of that DID (similar to mutual TLS authentication but with DIDs). This ensures that the agent you’re talking to is the one who owns the on-chain profile you looked up.
- Agents might also exchange certain credentials at this stage. For instance, if a client agent requires that any contractor agent has a certain minimum reputation or a specific certification, it can ask the other agent to present proof of that. The other agent could then send the relevant credential from the knowledge graph, or a zero-knowledge proof of its score meeting the threshold.

A2A’s messaging format could allow an initial “introductory message” where agents share such metadata. The protocol might remain agnostic of what’s inside (treating it as an opaque blob), but if both sides understand the convention, it works. Think of it like how HTTPS has certificate exchange – here agents exchange decentralized certificates.

Runtime Trust Queries

During an ongoing interaction, an agent can at any time query the registry if it needs updated info. For example, imagine Agent A has delegated a long task to Agent B. Partway through, new information emerges (perhaps someone logged a dispute about Agent B being unreliable). Agent A could periodically check Agent B’s profile or set up a subscription for changes. If Agent B’s reputation drops below a safe threshold or it gets flagged, Agent A might decide to halt the collaboration or ask for additional assurances. This dynamic monitoring is possible because the reputation system is live and transparent.

Moreover, agents can incorporate trust-based logic in their decision-making. For instance, an agent marketplace could automatically route a task to the highest AgentRank scorer available

in the relevant domain. Or an agent might decide how much information to share with a partner agent based on its trust score (high-trust partners get more access).

Trust Metadata in A2A Messages

We can embed lightweight trust metadata in *each message* header as well. For instance, an agent sending a message can include a short-lived token or reference that others can use to verify its reputation quickly. A concept could be a "reputation badge" – a signed token that says "Agent X's reputation > Y as of time T" signed by a reputation oracle. The receiving agent sees the badge, quickly checks the signature (perhaps the oracle's public key is known), and if valid, it accepts that the sending agent meets the criteria.

This way, the receiving agent doesn't have to query the whole graph every time it gets a message – it relies on the attached proof. To prevent misuse, these badges should be short-lived (with a timestamp and expiry) so that an agent whose reputation later falls cannot keep using an old badge.

In A2A terms, this could be implemented as an optional extension header in the message format, such as `X-Agent-Badge: <signed_token>`.

Interoperability and Composability

By integrating at the protocol level, we ensure **composability**: any agent platform or application that speaks A2A and wants trust features can seamlessly plug into the AgentRank system. For example, an AI workflow orchestrator could use A2A to let agents negotiate tasks, and use AgentRank to score bids or proposals from those agents (like how humans might use credit scores or seller ratings in marketplaces). The trust layer can also be composed with other layers – e.g., a decentralized **permission system** where an agent's on-chain reputation might grant it access to certain resources. One could imagine smart contracts that say "only agents with reputation > 0.9 can call this contract method" – integrating with A2A means agents prove their rep to the contract as well.

The complementary nature of A2A and a decentralized registry has been summarized by Intuition: *"A2A is the language and pipes; Intuition is the Yellow Pages and the credit score system"*[2]. Our integration strategy precisely realizes this vision: A2A messages carry pointers to the Yellow Pages (agent directory) and present credit scores (AgentRank) so that any agent-to-agent conversation occurs in the context of known identities and trust levels. This

greatly increases the safety and reliability of open multi-agent systems, as an agent can automatically be wary of messages from unknown or untrusted sources and prioritize interactions with reputable peers.

Evaluation Plan

We plan a comprehensive evaluation of the proposed AgentRank system along multiple dimensions: algorithmic performance, security (resistance to attacks), and usefulness in practical multi-agent tasks.

Simulation of Agent Networks

We will implement a simulator to create synthetic multi-agent ecosystems, where we can control the ground truth of agent behaviors (honest or malicious) and then test how well AgentRank identifies the trustworthy agents. The simulator will involve hundreds of agents with programmed interaction patterns:

- **Honest agents:** follow protocols, perform tasks correctly.
- **Malicious agents:** attempt to game the system – e.g., Sybil nodes (one entity controlling many agents that endorse each other), colluding groups that falsely upvote each other, or agents that perform well at first to gain reputation and then start betraying.
- **Mixed behavior:** some agents might oscillate or have occasional failures to test how forgiving/robust the system is.

We will simulate a sequence of time steps. At each step, agents interact (perform tasks together or endorse each other according to a scenario script), and we update the knowledge graph and run the AgentRank computation. We will measure:

- **Ranking quality:** How high do honest agents rank vs malicious ones? We expect honest ones to cluster at high AgentRank scores and malicious ones to be low. We can measure ROC curves if we consider “malicious” as the positive class to be ranked low. Ideally, the AUC (area under curve) for detecting bad actors by low score should be near 1.0.
- **Sybil containment:** We will specifically look at a scenario where an attacker inserts N Sybils. We measure the aggregate reputation that all those Sybils can accumulate relative to a single honest agent. With strong Sybil resistance, even as N grows, the total trust allocated to that Sybil cluster should remain low (bounded by what the attacker’s stake or

few entry points provide). This can be quantified by, say, the highest Sybil agent's R value or the sum of all Sybils' R.
- **Collusion resilience:** We simulate colluding groups and see if AgentRank unduly amplifies them. One metric is *reputation inflation factor* – how much higher do colluders rank by colluding than they would if they were isolated. We expect minimal inflation unless they had some real external trust.

We will also vary parameters like the damping factor α and any weight decay to see how it affects stability and fairness.

Empirical Deployment on Testnet

Beyond simulations, we aim to deploy a prototype of the system on a test network implementation of Intuition (for example, a testnet implementation of Intuition on Base Sepolia). We will register a number of agents and conduct tests of the on-chain components:

- Gas costs and performance of writing claims to the knowledge graph (for scalability evaluation).
- Latency of reputation updates (if computing off-chain and posting, or if using an on-chain simplified calculation for small N).
- We'll attempt some *realistic user testing* by inviting participants to play roles: e.g., have some people run agents that do useful tasks (maybe writing small essays or solving problems) and other people run agents that try to spam or cheat. The community (test participants) can then stake tokens (perhaps test tokens) to validate or refute claims, exercising the staking and dispute system.

The outcomes will help identify any practical issues, like: Is the dispute process too slow or too costly? Do participants understand how to use the staking mechanism (this touches on UX, which is not the focus here but important for real adoption)?

Security Analysis and Formal Verification

From a theoretical standpoint, we will analyze certain security properties:

- We aim to *prove* (or at least argue with bounds) some Sybil-resistance guarantees. For example, in a simplified model where there is a set of honest agents that fully trust each other and no trust to outsiders, and an attacker introduces m Sybils with no initial

outside trust, we can show that in the steady-state AgentRank scores, the total score of all Sybils is at most $(1 - \alpha)$ (coming solely from base), whereas honest agents share the rest (α) weighted by their trust network. This means Sybils can at best occupy $(1 - \alpha)$ fraction of the "trust space" no matter how many they are (and if α is high like 0.85, that leaves them only 15% at best, evenly split among possibly many identities)[6]. This would formally confirm the resistance.

- We will examine worst-case collusion: if one malicious agent does gain high trust (e.g., by actually doing good work to gain reputation, then turning bad), how much damage can it do by bringing others along? We might derive bounds that if an agent with rep R endorses k others fully, their scores will be at most R (it can't create more total trust out of nothing), and with damping, the system naturally limits multi-hop effects.
- Formal verification of smart contracts: The contracts for staking, vaults, and dispute should be audited and possibly formally verified for correctness (no double counting stake, proper slashing logic, etc.), to ensure the on-chain implementation truly reflects the intended model.

Evaluation of Privacy Features

For privacy, we can set up scenarios to test that essential functions work without exposing data:

- Use dummy zero-knowledge proofs to simulate how an agent might prove a statement about its reputation without full disclosure. For example, we might generate a SNARK that verifies an agent's score in a Merkle tree of scores. We would measure the proof generation and verification time to see if this is feasible in real-time agent interactions. This is more of a stretch goal, but it aligns with the privacy-preserving claim.
- We also analyze the graph for potential de-anonymization: although agents are pseudonymous, if one agent interacts heavily with another, someone might infer a relationship. Strictly speaking, that's metadata leakage that is inherent (like analyzing a public graph). Mitigations like rotating identities or not always using the same agent for everything can be discussed. Our evaluation might involve measuring how easily an outside observer could cluster agents by their interactions (using graph analysis), to ensure that if needed, agents can take measures to not reveal linkages they don't want revealed. This touches on a deep privacy topic and may be beyond initial implementation, but we note it.

Use-Case Demonstrations

Finally, to demonstrate *utility*, we will set up a few illustrative use-case scenarios:

1. **Collaborative Task Solving:** A complex task (say, a research problem) is broken into parts. Agents with different expertise must form teams. We will run this scenario with AgentRank enabled (agents prefer high-rep partners) and without it (agents choose randomly or by simple criteria) and compare outcomes. We expect that with AgentRank, successful teams form more often (since reliable agents cluster), yielding higher task completion rates.
2. **Open Agent Marketplace:** Suppose multiple agent service providers offer similar services. We simulate user agents picking a provider based on either round-robin or reputation-weighted selection. We then inject a malicious provider that sometimes fails or cheats. We should see that the reputation approach quickly down-ranks the bad provider, so user agents stop using it, whereas without reputation many users would continue to be harmed until perhaps a manual intervention.
3. **Trust Recovery Example:** We can illustrate how an agent that had a failure can recover. For instance, Agent Z has an early failure (maybe a dispute resolved against it, dropping its score). We then let it perform a series of good tasks which others endorse. We observe its AgentRank gradually improve due to decay of the bad event and influx of good endorsements. This shows that the system isn't one-and-done like a single rating, but a resilient measure that can adapt.

Throughout these evaluations, we will collect metrics and qualitative observations, which will be reported in a future full paper or technical report. The expectation is that AgentRank significantly enhances the robustness of multi-agent interactions: attacks that would derail a naive system (without a trust layer) are largely mitigated, and the overhead introduced (in terms of computation and communication) is manageable in practical settings.

Conclusion

We have presented **AgentRank**, a decentralized, trust-based framework for open multi-agent systems, designed to complement Google's A2A agent communication protocol. By leveraging a token-curated knowledge graph and a graph-based reputation algorithm, our approach addresses the critical question of *"Which agents should be trusted?"* in an environment where any agent can interact with any other. A2A solves the interoperability

problem by defining how agents talk; AgentRank (built atop Intuition's decentralized registry) solves the discovery and credibility problem by defining how agents *earn and convey trust*.

Our technical whitepaper introduced the context of A2A and highlighted the risk that without an open trust layer, the agent ecosystem could recentralize around proprietary directories or search engines. We argued that decentralization of the agent "phonebook" and reputation is essential to maintain a level playing field and to avoid dystopian outcomes where one company mediates all agent interactions. Drawing on Intuition's work, we described how a decentralized knowledge graph can serve as a **neutral meeting ground** for agents, where identities and claims are stored transparently and curated by the community. We then detailed AgentRank's design, providing formal notation for the trust graph and the iterative algorithm that propagates trust with damping (to ensure convergence and fairness) and incorporates time decay and dispute resolution to keep reputations accurate and up-to-date.

AgentRank is **decentralized and verifiable**: all inputs (endorsements, claims, stakes) are on a ledger, so anyone can audit why a certain agent has the reputation it does. It is **Sybil-resistant** by combining economic barriers (stake requirements, token curation) with trust graph analysis that limits the influence of isolated clusters of nodes[6]. It respects **privacy** by allowing pseudonymous participation and integrating with emerging cryptographic techniques for selective disclosure. And it is **expressive** enough to capture various signals about an agent: from formal credentials to peer feedback to observed performance metrics. These signals are distilled into a single (or multi-dimensional) AgentRank score that agents and humans can use as a basis for trust.

We also outlined how AgentRank could seamlessly integrate with the A2A protocol. As A2A-enabled agents handshake and advertise capabilities, they can reference their decentralized identity and even provide proofs of their on-chain reputation. This gives agents an automated way to **vet** each other before engaging in complex collaborations – effectively analogous to how web browsers check TLS certificates or how humans check reviews and ratings. By making this an open protocol interaction, we avoid any single point of failure: no central server is needed to broker trust, and no single authority can falsify or manipulate an agent's standing without consensus.

The broader implication of this work is the emergence of a **secure, open, and composable multi-agent coordination layer** for AI. In such a layer, new agents can join simply by creating a DID and proving themselves through actions; successful agents are recognized and sought

after through an open reputation system; malicious agents are quickly identified and isolated by the community. This fosters an ecosystem much like open-source software – meritocratic and transparent – as opposed to a closed platform. *The coordination framework for future machine intelligence should belong to everyone as an open commons, not be locked down by gatekeepers.* By marrying A2A's communication standard with a decentralized trust substrate, we move closer to that vision.

In future work, we plan to iterate on the AgentRank algorithm (e.g., exploring machine learning techniques on top of the graph to predict trustworthiness, or refining the weighting scheme with empirical data). We will also watch how the A2A protocol evolves in the wild and adapt our integration accordingly – possibly even contributing to the standard to ensure hooks for decentralized identity and reputation are included. We invite AI researchers, multi-agent system developers, and Web3 enthusiasts to collaborate on this endeavor. The challenges ahead are non-trivial (balancing robustness with openness is an ongoing process), but the reward is immense: a rich landscape of AI agents cooperating across organizational boundaries, where trust is managed as a public utility. With AgentRank and similar efforts, we aim to ensure the coming age of agent-based computing is one of **empowerment and collaboration for all**, rather than a rehash of centralized control .

In conclusion, AgentRank provides a promising foundation for scaling trust in multi-agent systems. It builds on proven ideas (like eigenvector-based reputation and token-curation) and tailors them to the unique context of AI agent ecosystems. Together with the A2A protocol, it enables an internet of autonomous agents that can not only communicate but also **coordinate safely**, knowing whom to trust. This lays critical groundwork for the future of distributed AI – potentially the substrate from which robust, decentralized artificial general intelligence might emerge, as countless specialized agents with aligned incentives work in concert to solve the world's complex problems.

References

1. Kamvar, S.D., Schlosser, M.T., & Garcia-Molina, H. (2003). *The EigenTrust Algorithm for Reputation Management in P2P Networks*. Proceedings of WWW 2003. (eigentrust.dvi)
2. 0xbilly (Billy Luedtke). (2025). *Agents of Change: Google's A2A and the Decentralized Future of AI Collaboration*. 0xIntuition on Medium, April 2025. (Agents of Change: Google's A2A and the Decentralized Future of AI Collaboration | by 0xbilly | 0xIntuition |

Apr, 2025 | Medium)

3. Google A2A Protocol Announcement and Documentation (2025). *Agent-to-Agent Protocol (A2A) – Open Standard for AI Agents*. [Online]. (Announcing the Agent2Agent Protocol (A2A)) (a2a-directory/docs/a2a-vs-mcp.md at main · sing1ee/a2a-directory · GitHub)
4. Intuition Whitepaper (2025). *Intuition: Decentralized Knowledge Graph and Trust Protocol*. (Whitepaper PDF, Intuition.systems) (Intuition Whitepaper)
5. **Intuition Medium Articles** (2025). Various blog posts by 0xIntuition team explaining the token-curated knowledge graph and trust approach (Intuition Medium).
6. EigenTrust & Related Trust Algorithms. (2003–2021). Research on decentralized reputation (EigenTrust, Advogato's trust metric, etc.) demonstrating power-iteration trust computation and Sybil resilience (eigentrust.dvi).
7. W3C. (2019). *Decentralized Identifiers (DIDs) v1.0*. W3C Recommendation. (Basis for agent decentralized identity in our system.)
8. W3C. (2019). *Verifiable Credentials Data Model 1.0*. W3C Recommendation. (Basis for credentials used in agent profiles and trust assertions.)

*(Note: The above references [2], [4], [5] refer to content summarized or quoted from Medium articles and whitepapers by the Intuition team and others, as cited inline. These provide context and statements used in this paper. Reference [1] is an academic citation for the EigenTrust algorithm. References [7] and [8] are standards underpinning parts of our design.