NON-HUMAN IDENTITY SERIES · PAPER #1

Agents Are Not Service Accounts

Why Non-Human Identity Needs Its Own Model

Mohamad Amin Hasbini

Independent researcher · Paris, France

Published April 21, 2026

ABSTRACT

> The service-account pattern is structurally wrong for AI agents. Per-agent identity with rotation, revocation, and attestation is the operational minimum.

KEYWORDS — AI agent security, non-human identity, post-quantum cryptography, agent authorization, capability tokens, NIS2, DORA, EU AI Act

---

CITE AS

Hasbini, M. A. (2026). *Agents Are Not Service Accounts*: Why Non-Human Identity Needs Its Own Model. Non-Human Identity Series, Paper #1.
Available at https://mahasbini.org/papers/01-agents-not-service-accounts/
PDF https://mahasbini.org/publications/papers/01-agents-not-service-accounts.pdf

In conversations with French enterprise CISOs over the last year, across banking, defense, and industrials, the pattern is consistent. AI agents run on shared service accounts.

Not as an edge case. As the default.

The pattern varies superficially. A shared OpenAI key read by a CRM agent, a procurement agent, and a ticketing agent. An AWS IAM role attached to a container hosting three different LangChain pipelines. A single Azure AD application registration acting as the identity for an entire "AI team."

The structural error is identical. The organization treats the agent as if it were a cron job, a batch process, or a microservice. It is none of those things.

And the service-account pattern that worked, more or less, for the first three fails visibly for the fourth.

This piece is an attempt to name the failure and describe what replaces it.

This paper addresses **ASI03 (Identity & Privilege Abuse)** from the OWASP Top 10 for Agentic Applications 2026: the structural identity gap. **ASI07 (Inter-Agent Communication)** is companion Paper #3 territory. **ASI02 (Tool Misuse)** is Paper #2. The rest of the OWASP taxonomy (goal hijack, memory poisoning, cascading failures) lives at layers above and adjacent to identity. Per-agent identity is the minimum floor, not the whole building.

Why service accounts existed in the first place

The service account is a 1990s answer to a 1990s question. How does a scheduled process authenticate without a human typing a password?

The answer was a non-interactive identity with static credentials: a username and password, later an SSH key, a Kerberos keytab, an X.509 cert, an OAuth client secret. The security model made three quiet assumptions.

The actor is a program that executes a fixed set of operations in a fixed order. The credential is long-lived because rotation is expensive. Scope is coarse because the program will only ever do what it is coded to do.

Twenty-five years later, the workload-identity conversation has moved on. SPIFFE. Workload identity federation. OIDC client assertions. Zero trust as articulated in NIST SP 800-207. Short-lived, narrow-scoped, attestable, auto-rotated.

But the language and the tooling of the service account persisted into the LLM-agent era. Most 2023-2025 agent frameworks treat "the API key" as a deployment detail.

That casualness is the inherited artifact of two decades of static-credential thinking. It is wrong for agents. Here is why.

Four structural reasons the pattern fails

The actor is non-deterministic. A service account executes a known set of API calls in a known order. A LangChain agent chooses its tool invocations at runtime from a surface of a dozen or a hundred functions, driven by an LLM whose reasoning is not reproducible across runs. The security model that assumed “this process will only ever call these five endpoints” is wrong by construction. Static least-privilege policies written against the call graph no longer exist because the call graph no longer exists.

The delegation chain is deep. A user invokes an agent. The agent invokes another agent. The second agent calls a tool. The tool calls a downstream service belonging to a third party. Each hop should carry a traceable, revocable identity, what enterprise IT calls **On-Behalf-Of (OBO)** and what **RFC 8693 (OAuth Token Exchange)** formalizes. Shared service accounts erase that chain. The audit log collapses at exactly the layer where regulators now want granularity.

The blast radius is asymmetric. An attacker who injects a malicious instruction into a document, an email, or a shared workspace can cause an agent to act against its principal using legitimate credentials. OWASP LLM Top 10 codifies this under Excessive Agency and Insecure Output Handling. The attacker does not need to steal the credential. The attacker needs the credential to be shared, coarse-scoped, and attached to an agent whose instruction surface is adversarially reachable.

Simon Willison calls this configuration the “lethal trifecta”: private data access, exposure to untrusted content, and outbound communication capability combined. The service-account pattern is what makes the trifecta enterprise-scale: a shared credential aggregates access that, under prompt injection, funnels into a single exfiltration event. Per-agent credentials do not prevent the trifecta; they limit the blast radius to one agent’s scope instead of a whole team’s.

Regulators do not recognize the pattern. NIS2 Article 21 requires cryptographic controls and access management commensurate with risk. DORA requires auditable ICT access trails. The EU AI Act requires high-risk AI systems to be logged, overseen, robust, and documented per Annex IV. A shared team credential cannot produce per-agent logging. It cannot demonstrate per-agent human oversight. It cannot be individually decommissioned. “A team’s shared OpenAI key” satisfies none of the three regimes.

ACPR and Banque de France inspection notes from late 2025 and early 2026 have started to include AI-agent access control among items reviewed during ICT on-site missions. ANSSI audit guidance for NIS2-qualified products treats shared non-human credentials as a material control weakness. The direction of travel is unambiguous. The pattern that was tolerated as a pragmatic shortcut in 2023-2024 will not survive the 2026-2027 supervisory cycle.

What per-agent identity actually means

“Per-agent identity” sounds obvious and collapses on contact with engineering reality. A workable version has six properties.

Unique cryptographic identity. Each agent has its own key pair or verifiable credential. The private material is not shared.

Lifecycle binding. Registration, attestation, rotation, revocation, each logged.

Rotation on a schedule proportional to risk. 30-90 days for low-risk summarization agents. Per-session for agents with access to financial systems.

Revocation that actually takes effect in minutes, not days.

Attestation that survives verification: the agent's owner, risk tier, declared capabilities, code version, platform.

Discoverability through a registry. One authoritative list of every agent in production. Most organizations in April 2026 cannot produce this list in under two weeks. That is the first and most urgent finding.

Three existing models, their strengths and their gaps

SPIFFE / SPIRE. Short-lived credentials by default. Strong platform-attestation. The per-workload model maps onto per-agent. The gap is that SPIFFE pins identity to a process on a host, while agents often run inside shared orchestration runtimes serving many logical agents. Attaching one SVID per logical agent requires additional plumbing.

Workload identity federation. Eliminates static cloud credentials in external systems. Trust flows from the platform hosting the agent, not the agent itself. If the platform is shared across many agents, the granularity gap returns one level up the stack.

OIDC client assertions (RFC 7523). Asymmetric credentials, no shared secrets, standardized. Naturally supports per-agent key pairs. The gap is operational: the tooling around bulk issuance, automated rotation, and revocation at the per-agent scale is underdeveloped in most commercial identity providers.

None is a drop-in fit. All three cover the credential and the channel. None covers the agent-specific semantics: the capability metadata, the delegation chain, the auditable call graph.

What's missing, and what to build in 2026-2028

The 2026-2028 identity layer for AI agents needs to add five operational elements that existing workload-identity models do not cover.

An agent-aware registry with an API, queryable by the SOC and the audit function.

Delegation semantics in the token, using OAuth Token Exchange (RFC 8693) plus conventions for task-scoped delegation.

Capability-based authorization policy that expresses "Agent X may read procurement records belonging to user Y for the duration of a single task that user Y explicitly initiated."

An auditable call graph that answers "which agent did what, on whose behalf, with what reasoning abstract" months later.

PQC-ready transport on MCP and A2A channels. Hybrid key exchange today, because the data these channels carry has a useful life that outruns the classical-adversary assumption.

None of the five requires a new product purchase. All can be started in a single quarter.

Five things your CISO can do next Monday

One. Inventory your agents: every agent in production, pilot, and on developers' laptops talking to production APIs. If you cannot produce this list in two weeks, that is your first audit finding.

Two. Decommission the shared API key. Per-agent credentials, even if the first pass is a per-agent OAuth client registration rather than a full SPIFFE rollout.

Three. Name a single owner for agent identity issuance. Not a committee. One person, one function.

Four. Wire agent identity into existing IAM tooling. Do not build a parallel stack. Agents belong in the same directory as humans for joiner-mover-leaver.

Five. Add MCP and A2A channels to the PQC migration scope. This is free if a TLS inventory is already in flight. It becomes a separate program if it is not.

None of the five is being done consistently in the French enterprises I audit in April 2026. The gap between "could be done Monday" and "is being done" is the operational content of the next year of CISO work on non-human identity.

Closing

Service accounts were a good answer to "how does a cron job authenticate." They are the wrong answer to "how does an AI agent that acts on behalf of humans, across deep delegation chains, using non-deterministic tool selection, under adversarial prompt-injection pressure, authenticate itself to the rest of the enterprise."

The industry will eventually ship purpose-built non-human identity platforms, and most CISOs will eventually buy one. In the meantime, the five Monday actions above are what responsible agent operation looks like with the tools available today.

This is paper #1 of a five-part series on non-human identity. Paper #2, *Authorization for AI Agents: Beyond RBAC*, tackles the next layer up. Once you have per-agent identity, the question is what that identity is permitted to do, and how the policy is evaluated in real time against the intersection of agent permissions, user delegation, and task scope.

Paper #3 goes deeper on agent-to-agent communication, including zero-knowledge proofs (ZKP) that let an agent prove authorization without revealing its credential, bounding the blast radius of the lethal trifecta.

Companion tools

- **Agent Identity Platform**: interactive reference implementation of the three-pillar governance flow. Register agents, create policies, send messages with full audit trail.
- **AI Agent Security Maturity Assessment**: diagnostic that scores an organization against the three-pillar framework. Executive-ready output for board conversations.
- **AgentTrustLab**: working simulator for agent-to-agent authentication under PQC and ZKP constraints (Paper #3 territory).

Feedback welcome. Field observations shape what comes next.

Appendix: Concrete construction pointers for peer reviewers

The body of this paper is written for CISO and board audiences. For cryptographic peer reviewers, the following pointers name the specific constructions behind "per-agent identity" and "PQC-ready transport" referenced above.

Per-agent credential. A candidate scheme: an SD-JWT-VC issued by the organization's agent registry, signed with ML-DSA-65 (NIST FIPS 204, finalized August 2024), bound to a SPIFFE SVID that anchors the transport channel identity. Revocation via status list. Lifecycle events logged to the registry.

Delegation. RFC 8693 OAuth Token Exchange with ML-DSA-signed actor claims in the `act` field. Task-scoped `scope` values. Transient delegation tokens rather than long-lived refresh tokens.

Transport (MCP, A2A). Hybrid X25519+ML-KEM-768 key encapsulation per the current IETF hybrid KEM drafts. Hybrid ML-DSA-65+ECDSA signatures on certificate chains during the FIPS 203/204/205 migration window per NIST post-quantum migration guidance.

RFC 7523 PQC implications. Client assertions currently use RS256 or ES256. Migration path: accept ML-DSA-65 as an `alg` value, update JWK `kty` to NIST-registered ML-DSA key types, support hybrid assertions during the window where relying parties are mixed.

A full threat model, concrete construction, and security argument is deferred to a companion technical note. Peer review welcome before publication.

About the author

Amin Hasbini is an AI and cybersecurity executive based in Paris. Former director of Kaspersky's Global Research & Analysis Team (GReAT) for the Middle East, Turkey, and Africa. Twelve years, seventy countries of threat coverage. Invited contributor to the French Senate's OPECST report on AI risks (2024). Former subject-matter expert on ICANN's second DNS Security, Stability, and Resiliency Review Team (SSR2, 2017–2019). Current focus: post-quantum cryptography maturity and AI agent security inside regulated enterprises. mahasbini.org.