arXiv:2604.03733v1 [q-fin.GN] 4 Apr 2026

SoK: Blockchain Agent-to-Agent Payments

Yuanzhe Zhang[1], Yuexin Xiang[2], Yuchen Lei[3], Qin Wang[4], Tian Qiu[1], Yujing Sun[1], Spiridon Zarkov[5], Tsz Hon Yuen[2], Andreas Deppeler[6], Jiangshan Yu[7], and Kwok-Yan Lam[1]

[1] Digital Trust Center, Nanyang Technological University, Singapore
[2] Faculty of Information Technology, Monash University, Australia
[3] School of Cyber Science and Engineering, Wuhan University, China
[4] CSIRO Data61, Australia
[5] Zark Lab, Singapore
[6] School of Business, Monash University, Malaysia
[7] School of Computer Science, University of Sydney, Australia

Abstract. Agentic AI rivals human capabilities across a wide range of domains. Looking ahead, it is foreseeable that AI agents will autonomously handle complex workflows and interactions. Early prototypes of this paradigm are emerging, e.g., OpenClaw and Moltbook, signaling a shift toward Agent-to-Agent (A2A) ecosystems. However, despite these promising blueprints, critical trust and security challenges remain, particularly in scenarios involving financial transactions. Ensuring secure and reliable payment mechanisms between unknown and untrusted agents is crucial to complete a fully functional and trustworthy A2A ecosystem. Although blockchain-based infrastructures provide a natural foundation for this setting, via programmable settlement, transparent accounting, and open interoperability, trust and security challenges have not yet been fully addressed. Hence, for the first time, we systematize blockchain-based A2A payments, e.g., X402, with a four-stage lifecycle: discovery, authorization, execution, and accounting. We categorize representative designs at each stage and identify key challenges, including weak intent binding, misuse under valid authorization, payment–service decoupling, and limited accountability. We highlight future directions for strengthening cross-stage consistency, enabling behavior-aware control, and supporting compositional payment workflows across agents and systems.

Keywords: LLM agent · Agent-to-agent · Blockchain · Payment · X402

1 Introduction

AI agents are evolving from one-off chatbot interactions into persistent software entities that coordinate across tools, services, and agent-to-agent environments at scale [1]. Emerging agentic systems such as OpenClaw [2, 3] and agent-native platforms like Moltbook [4] illustrate this transition toward ecosystems in which agents interact continuously with external services and other agents.

More broadly, large language models (LLMs) have enabled agentic systems capable of planning, reasoning, and invoking external tools to carry out tasks over extended periods of time [5–12]. These agents act on behalf of human users and organizations to access APIs, acquire data or computational resources, negotiate with other agents, and complete service workflows with limited human intervention [13, 14]. As agent operation becomes persistent rather than episodic, payment capabilities become necessary, as coordinating access to resources, services, and other agents inherently requires economic mechanisms [15, 16].

However, conventional human-oriented payment infrastructures require payments to be explicitly initiated and authorized by human users, typically through predefined merchant interfaces (e.g., checkout flows or API endpoints) [17–19]. This model does not extend naturally to agent ecosystems, where agents autonomously discover services, compose multi-step workflows, and interact with previously unseen counterparties. Such agent-to-agent interactions instead require payments to be programmable, interoperable across heterogeneous platforms, and verifiable without bilateral trust, while supporting high-frequency, low-value interactions across services and counterparties. Recent extensions to conventional payment infrastructures, such as Mastercard Agent Pay [20], enable AI agents to transact using tokenized payment credentials under delegated authorization constraints. However, as they remain tied to card-network authorization and settlement rails, each interaction is processed as an independent authorization, making such systems ill-suited for workloads involving frequent small payments or multi-step tasks due to repeated authorization overhead. Hence, to support and enable a fully automated agent-to-agent (A2A) payment system, blockchain-based infrastructures offer a compelling solution. With their programmability, executability, global accessibility, and inherent transaction verifiability, such systems make it possible to embed value transfer directly into automated workflows, eliminating the need for custom bilateral integrations and reducing dependence on fully trusted intermediaries.

We notice that recent blockchain-based systems and design proposals [21–23, 14, 24, 25] were connecting LLM-enabled agents to on-chain or chain-anchored payment rails, supporting automated settlement, metering, and basic accounting for service interactions. But unfortunately, enabling agents to pay autonomously does not by itself ensure that payment intent, authorization, execution, and service outcome remain aligned [26, 27]. Once financial authority is delegated to autonomous agents, errors, misalignment, or adversarial manipulation can lead directly to financial loss, unauthorized spending, and violations of governance constraints [28–31, 26]. Recent evidence further shows that agent protocol stacks already expose protocol-logic vulnerabilities and supply-chain style attacks [32], while payment-enabled workflows amplify the consequences of prompt injection and interaction manipulation [33, 34]. At the same time, blockchain payment infrastructures introduce practical constraints related to latency, fees, scalability, and privacy that must still be reconciled with off-chain service execution and provenance tracking [35].

Therefore, in this Systematization of Knowledge (SoK) paper, we, for the first time, systematically examine and evaluate the trust, privacy, and security risks of existing mechanisms for Agent-to-Agent payment systems, particularly in which agents can autonomously or conditionally initiate payments, receive payment-triggered services, or both through blockchain-based infrastructures. Particularly, we categorize the identified risks and challenges, including weak intent binding, misuse under valid authorization, payment–service decoupling,and limited accountability, into a four-stage life cycle, spanning discovery, authorization, execution, and accounting. Our contributions are summarized as follows:

- We propose a lifecycle model for blockchain-based payments for AI agents, which provides a common abstraction for reasoning about discovery, authorization, execution, and accounting across heterogeneous systems.
- We organize and systematize the emerging design space by mapping representative mechanisms, system assumptions, and deployment patterns onto this reference model, thereby clarifying how current approaches differ and where their trade-offs arise.
- We derive a structured view of the risk surface and research gaps in agentic payments, highlighting challenges in delegated spend control, service–payment coupling, accountability, privacy, scalability, and compliance.
- We point out future directions and possibilities to address the raised issues.

2 Preliminaries

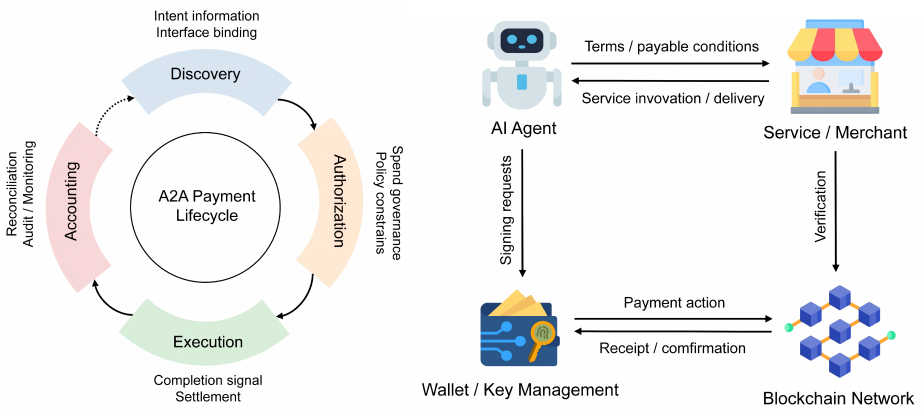
We distinguish three core abstractions underlying agent-mediated payment systems: large language models (LLMs) as generative components, AI agents as systems that interpret model outputs and execute external actions, and blockchain as the programmable settlement substrate for realizing payment logic [36–41].

2.1 LLMs and AI Agents

LLMs are generative models, typically based on Transformer architectures [36], that produce text and intermediate reasoning traces conditioned on context [37]. In this paper, the LLM is treated as a component that generates candidate actions and structured outputs. We view AI agents as software systems that orchestrate model outputs, memory, and tool execution to perform multi-step interactions with external services [38, 39, 42, 43]. A typical interaction loop consists of generating action directives, executing tool calls, and incorporating observations into subsequent steps. In the payment setting, this reduces to sequences of tool-mediated interactions through which model outputs are translated into payment-relevant operations.

2.2 Blockchain and Programmable Payment

Blockchains provide a decentralized, append-only settlement layer for transferring value and verifying system state without central intermediaries [40, 44].



(a) Agentic Payment Lifecycle (b) Participant interaction framework

Fig. 1: Overview of blockchain A2A payments

Smart-contract platforms such as Ethereum extend this model to programmable transactions, enabling payment conditions, delegation constraints, and execution rules to be enforced directly within the settlement process [41]. Recent advances in scaling (e.g., sharding and L2 systems) have reduced latency and transaction costs [45–47], improving the practicality of high-frequency payments. Accordingly, we treat blockchain as the infrastructure that enables verifiable and programmable settlement for agent-mediated interactions, while introducing trade-offs in usability, recourse, and risk allocation [48].

3 Agent-to-Agent Payment Lifecycle

We summarize blockchain-based agent payments using a four-stage lifecycle (Fig. 1a), capturing recurring functional stages across existing systems. This lifecycle serves as a compact analytical reference for structuring the survey and localizing mechanisms and risks without committing to a specific architecture. Complementarily, Fig. 1b illustrates the main participants and interactions: payable conditions arise at the agent–merchant interface, payment actions are executed via wallet or key-management components, and settlement occurs on the blockchain, with on-chain receipts supporting confirmation and reconciliation.

❶ **Discovery.** The lifecycle begins with identifying a payable operation and constructing a structured payment intent that binds the payment obligation to its execution context, including the requested resource, applicable terms, and a stable request identifier. This representation enables subsequent stages to reference a persistent payment obligation and coordinate execution, retries, and accounting.

❷ **Authorization.** The system determines whether an authenticated payment request is admissible under predefined governance rules. Delegated spend-

ing constraints, such as budget limits, policy controls, and approval requirements, are enforced over the authenticated principal to bound permissible execution. The outcome is an authorization decision that governs how the payment intent may be executed.

❸ **Execution.** Given authorization, the system realizes the payment intent as a concrete payment action and produces a completion signal that allows the workflow to progress. This stage includes handling transaction outcomes such as failures, retries, and settlement delays.

❹ **Accounting.** The system maintains records that link payment actions with their originating intents and observed outcomes to support reconciliation, auditability, and operational monitoring. This stage provides traceability across the workflow and enables downstream processes such as billing verification and dispute resolution.

The lifecycle separates four fundamental functions, and we use this reference model throughout the SoK to align surveyed approaches and localize risks, limitations, and open challenges to their corresponding stages.

4 Existing Approaches by Lifecycle Stages

We analyze existing approaches from a lifecycle perspective. Table 1 maps representative agentic payment systems to the stages they primarily support, motivating a stage-wise decomposition.

4.1 Discovery and Intent Binding

The discovery stage determines how agents locate payable counterparties and construct payment intents tied to a specific interaction context. We organize this stage along two dimensions: (i) the *discovery substrate*, which defines how agents and services are resolved, and (ii) the *intent-binding primitive*, which specifies how payment obligations are associated with the interaction context.

Discovery Substrates (D): Discovery mechanisms differ in how counterparties and service metadata are resolved. We distinguish three dimensions: descriptor-based, identity-augmented, and behavior-derived discovery.

D1. Descriptor-based discovery. Services expose structured capability descriptors (e.g., AgentCards [49]) that specify supported actions, invocation schemas, and potentially pricing or payment requirements. These descriptors may be retrieved directly from service endpoints (e.g., HTTP-accessible metadata) or indirectly via registry-based resolution, where identifiers or descriptor references are anchored on-chain and resolved through smart contracts or indexing services [21].

D2. Identity-augmented discovery. Identity-centric systems enrich discovery with identity and delegation information, such as relationships among users, agents, and sessions. They may also use trust-related signals like claims, stake, or past behavior (e.g., Kite [23], EIP-8004-style mechanisms [50]). Skyfire is a concrete example of this approach. Its *Know Your Agent (KYA)* layer and service

Table 1: Lifecycle decomposition of existing approaches by stages they primarily support. ■ primary contribution; ▨ discussed; □ not covered.

Paradigm	Designs	Disc.	Auth.	Exec.	Acct.
Service-Specified Payment	x402 [53]	■	▨	■	▨
	MPP [54]	■	▨	■	▨
	Vaziry *et al.* [21]	■	▨	■	▨
	Skyfire [51]	■	▨	■	▨
	Agent Pay [20]	□	▨	■	▨
Owner-Specified Payment	ERC-4337 [55]	□	■	▨	□
	Dimitrov *et al.* [56]	□	■	▨	▨
	Gorzny *et al.* [57]	□	■	▨	▨
Identity-Augmented Payment	Kite [23]	▨	■	■	▨
	EIP-8004 [50]	■	▨	□	□
Payment Execution Infrastructure	Nanopayments [58]	□	▨	■	□
	Tempo [59]	□	□	■	□
	Ravi *et al.* [60]	□	▨	■	□
Cross-Stage Coupling	Li *et al.* [26]	□	□	■	■
	Acharya *et al.* [27]	▨	■	▨	■

directory link agent identities and credentials to discoverable services, and bring identity verification into both interaction and payment workflows [51].

D3. Behavior-derived discovery. Discovery can also be driven by behavioral signals observed from prior interactions rather than declared capabilities or semantic descriptions. In this paradigm [52], services are ranked based on interaction traces such as payment flows, where transactions act as implicit endorsements, and reputation is propagated according to the quality of participating agents, transaction value, and temporal recency.

Intent-Binding Primitives (B): Intent-binding mechanisms differ in what elements of an interaction a payment obligation is associated with, such as individual service requests or identity and session context.

B1. Payment-term binding via in-band signaling. In x402-style interaction flows [53, 21, 54], services respond to requests with payment requirements (e.g., asset type, recipient address, amount) as part of a challenge–response exchange. Agents satisfy these requirements by attaching a corresponding payment transaction or proof in subsequent requests, thereby completing the interaction.

B2. Context and provenance binding. Identity-centric designs bind payment authorization to identity and delegation context, such as user identity, agent instance, and session-level constraints (e.g., Kite [23]). In these systems, payment intents are realized as context-scoped delegation objects whose validity and interpretation depend on the associated identity and session state, enabling actions to be attributed to specific principals and delegated sessions. More expressive designs make this binding explicit by representing user intent as verifiable objects. For example, Acharya [27] introduces on-chain intent proofs that bind user authorization to payment execution, allowing transactions to be verified against user-approved constraints.

4.2 Authorization

Authorization builds on intent binding by enforcing which payment intents may be executed and under what constraints. It operates over requests that are bound to a principal through authentication (e.g., cryptographic signatures, credentials, or session context), and determines whether the resulting payment intent is admissible under delegated policies. We organize existing designs along two dimensions: (i) *authorization carriers* (A), which define where and how spending authority is enforced, and (ii) *policy expressiveness* (E), which captures how richly constraints over spending behavior are specified.

Authorization Carriers (A): Authorization carriers differ in where spending authority is enforced along the transaction path, such as at contract invocation, asset transfer, or wallet-level validation prior to execution.

A1. Contract-mediated delegation. Agent actions are mediated by smart contracts that act as execution intermediaries and enforce access control over callable functions (e.g., role-based permissions) [61]. Agents invoke contract functions under predefined roles, and the contract determines whether the invocation is permitted based on its internal logic.

A2. Allowance- and approval-mediated spending. Spending authority is delegated via token allowances or approval mechanisms, where agents are permitted to transfer assets within predefined limits (e.g., ERC-20 approvals). Authorization is enforced at the asset layer, where transactions are validated against allowance constraints such as maximum spendable amount or approved spender. Authentication is provided by signature-based transaction validation, either through on-chain approval transactions or off-chain signed permits (e.g., EIP-2612 [62]), binding the transfer request to the token holder or an approved spender address.

A3. Wallet-mediated programmable authorization. Account abstraction (AA), as instantiated in ERC-4337 [55, 63], shifts authorization logic into smart contract wallets that validate user operations prior to execution. Each operation is submitted as a *UserOperation* and processed by an entry point contract, which invokes wallet-defined validation logic to check signatures, policies, and contextual constraints before inclusion. Such programmable authorization has been explored in both wallet systems and agent-driven designs (e.g., [56, 57]), where validation logic is extended to support automation and policy enforcement.

Policy Expressiveness (E): Policy expressiveness captures what constraints can be specified over delegated spending behavior, ranging from access-level permissions to transaction-level bounds and contextual policies.

E1. Access-level constraints. Policies specify which actions or contract functions an agent is permitted to invoke (e.g., role-based access control) [61]. Constraints are defined over callable operations, determining whether a given invocation is allowed.

E2. Transaction-level constraints. These constraints are typically enforced through token allowance or approval mechanisms (e.g., ERC-20 approvals), where transactions are validated against predefined limits at execution time. More flexible forms of transaction-level constraints can be expressed through signature-

based approvals (e.g., EIP-2612 [62] and Permit2 [64]), enabling fine-grained control over transaction parameters.

E3. Contextual and stateful policies. Policies incorporate contextual information and state-dependent conditions, such as rate limits, cumulative spending bounds, or delegation scopes, evaluated during validation (e.g., in AA-based systems) [56, 57]. Constraints may depend on interaction context, temporal conditions, or historical state.

4.3 Execution and Settlement

The execution stage concerns how payment intents are realized as concrete transactions, how these transactions are submitted and validated, and what conditions define completion for subsequent workflow progression. We organize this stage along three dimensions: (i) *settlement paths* (S), which determine how and where payments are finalized; (ii) *submission and fee orchestration* (O), which determines how transactions are constructed, submitted, and funded; and (iii) *access-gating evidence* (G), which determines what observable signals are used to trigger service access or workflow continuation.

Settlement Paths (S): Settlement paths differ in where payment state is maintained and when settlement is finalized, such as through direct on-chain inclusion or off-chain coordination with deferred settlement.

S1. Direct on-chain settlement. Each interaction is realized as an on-chain transaction, with completion defined by transaction inclusion in the blockchain [53, 21, 18]. Agents construct and submit transactions that directly transfer assets to the service provider, and subsequent workflow progression is conditioned on confirmation of inclusion. The mechanism operates at the blockchain layer, where the payment state is recorded and finalized on-chain.

S2. Off-chain–coordinated settlement. Payment interactions are executed off-chain through signed updates or bilateral agreements, with final settlement deferred to a later on-chain transaction [23, 59]. Agents exchange off-chain payment updates that represent incremental transfers, which are later consolidated and settled on-chain as a single transaction. The mechanism operates across off-chain coordination and on-chain settlement layers, where the payment state evolves off-chain and is periodically committed on-chain.

Submission and Fee Orchestration (O): Submission and fee orchestration mechanisms differ in who submits the payment transaction and how execution costs are provisioned. We distinguish three patterns: direct client submission, account-abstraction-mediated relaying, and facilitator-mediated submission based on off-chain authorization.

O1. Client-submitted transactions. Agents construct, sign, and submit transactions directly to the network and are responsible for paying transaction fees. Execution is tied to on-chain confirmation, with agents managing nonce, gas pricing, and submission timing [21].

O2. Account-abstraction mediated submission. Account abstraction (e.g., ERC-4337 [63]) introduces an alternative submission flow in which agents produce

UserOperations relayed by bundlers and executed via an entry point contract [55–57]. Paymasters may sponsor transaction fees on behalf of agents, decoupling fee payment from the originating account.

O3. Facilitated submission with off-chain authorization. Agents authorize token transfers off-chain using signed transfer approvals (e.g., EIP-2612 [62], or Permit2 [64]), and a facilitator submits the corresponding on-chain transaction on their behalf [65]. facilitators handle transaction broadcast and may sponsor gas, while recovering costs through the payment flow or application-level fee handling.

Access-Gating Evidence (G): Access-gating mechanisms differ in what observable evidence is required to trigger service execution or workflow continuation, such as on-chain transaction inclusion or off-chain payment state.

G1. Inclusion-gated access. Service access is conditioned on the inclusion of a corresponding on-chain payment transaction [53, 21]. Agents submit a payment transaction, and services verify its inclusion before proceeding with execution. The mechanism operates at the blockchain layer, where access is gated by confirmed on-chain payment records. Recent industry protocols such as Stripe's Machine Payments Protocol (MPP) follow a similar model, integrating payment authorization and confirmation into API interaction flows and gating service access on successful payment completion [54]. Such interaction patterns may be deployed over different execution environments, including emerging infrastructures such as Tempo that support crypto-based agent payments [59].

G2. Off-chain update–gated access. Access is granted based on off-chain payment state or deferred-settlement updates, rather than direct on-chain transaction inclusion. Services accept intermediate payment evidence as sufficient to continue execution, while final settlement is completed later on-chain. Such designs arise in deferred-settlement or liquidity-aware payment protocols [60].

G3. Proof-based or attestation-gated access. Access is granted based on verifiable payment evidence beyond direct transaction inclusion, such as signed receipts, cryptographic proofs, or third-party attestations. Agents present verifiable artifacts that attest to payment completion or entitlement, which services validate before execution. Such designs are explored in systems that couple payment with verifiable execution or authorization evidence (e.g., A402 [26], Acharya [27]), where access decisions are derived from cryptographically verifiable proofs rather than on-chain inclusion alone.

4.4 Accounting

The accounting stage concerns how payments are verified and linked to service outcomes. We structure this in two ways: (i) *verification evidence* (V), which defines what constitutes valid evidence of payment, and (ii) *service–payment coupling* (C), which defines how payments are associated with outcomes.

Verification Evidence (V): Verification evidence differs in what artifacts are used to establish that a payment has occurred, and at which layer such evidence is generated and validated.

V1. On-chain transaction evidence. Blockchain transaction records serve as the primary proof of payment, with completion defined by inclusion in the ledger [53, 21, 66]. Agents or services verify that a transaction transferring the specified asset and amount to a designated recipient has been included on-chain.

V2. Interaction-level receipts and logs. Systems may generate application-level artifacts such as execution logs, interaction traces, or service-issued receipts that record the occurrence of payment-related events. Such artifacts are commonly used for auditing and debugging in agent-based systems, where execution traces or policy decision records may be retained for verification or analysis [66].

Service-Payment Coupling (C): Service-payment coupling mechanisms differ in the timing and strength of linkage between payment and service execution, ranging from execution-triggered access, to post hoc accountability, and to protocol-level enforcement.

C1. Execution-trigger coupling. Payment confirmation is used as a condition to trigger service execution, establishing a direct dependency between payment events and invocation of service logic [53, 21, 18].

C2. Post hoc accountability linkage. Mechanisms such as insurance-based accountability layers associate payments with service outcomes after execution through retrospective evaluation and dispute resolution processes [67]. These mechanisms support auditing and responsibility attribution by linking payment events to outcomes ex post, rather than enforcing coupling during execution.

C3. Enforcement-based coupling. Payment settlement and service execution are cryptographically or protocol-level interdependent, such that neither can be completed unilaterally. For example, A402 [26] introduces atomic service channels that bind payment completion to service delivery via adaptor signatures, ensuring that payment is finalized only upon the release of execution-dependent secrets.

5 A2A Payment Challenges

We have analyzed risks and limitations above by mapping them into the stages of the agent payment lifecycle, providing a structured view of where vulnerabilities and operational challenges arise across different phases of interaction. We then further distinguish technical weaknesses, transaction level mismatches, and broader governance constraints in each stage and discuss separately on agent authentication, including identity-related issues, and future directions.

5.1 Discovery and Intent Binding Risks

Discovery and intent binding use exposed metadata to produce payment intents. However, neither of them guarantees correct execution or institutional admissibility.

Technical level. Discovery via endpoints or registries (D1) constructs payment intents from externally exposed metadata without authenticating semantic identity [13, 68]. This creates a phishing-like surface where adversaries expose

valid-looking descriptors through counterfeit endpoints, typosquatted domains, spoofed Agent Cards, or deceptive registry entries [69–72]. As a result, agents may form syntactically valid payment intents for malicious or nonexistent services, binding payments to the wrong counterparty.

Transaction level. Intent-binding mechanisms, for example, in-band payment signaling (B1), embed payment conditions in the request. However, they do not reliably tie payments to a specific session, a concrete outcome, or verifiable fulfillment. Acharya [27] addresses this issue by proposing a stronger design based on decentralized identity, on-chain intent proofs, and attested execution, but it remains a proposal and has not been deployed in mainstream systems. As a result, a payment may still be valid even when the interaction is replayed, mismatched, or produces missing, partial, or incorrect service outcomes.

Legal and institutional level. Discovery selects counterparties based on reachability and functional compatibility. However, it neither checks regulation requirements nor institutional admissibility, such as jurisdiction, sanctions, licensing, or platform policies. Such checks are rarely performed before intent formation, which prevents service resolution from governance constraints emphasized in agent economies and interoperable systems [73–75]. As a result, agents may execute technically valid payments to counterparties, but they are institutionally prohibited.

> **Insight.** Discovery anchors payment to externally described metadata rather than execution semantics or admissibility. It may lead to errors happening at this stage, which would propagate downstream, such that later stages may consistently execute and record payments that are internally valid yet fundamentally misaligned with the intended interaction.

5.2 Delegated Authorization and Spend Control Risks

Authorization enforces transaction-level admissibility under delegated control, but does not capture the broader behavioral meaning of spending over time.

Technical level. Authorization validates transactions at submission but assumes that transaction generation is trustworthy. If the agent or its environment is compromised through prompt injection [76], model manipulation [77], key leakage [78], software vulnerabilities [79], or social engineering [80], adversaries may generate transactions that satisfy authorization policies. Because delegated authority is encoded as persistent rules with reactive revocation, policies that are initially valid may become unsafe in operation, but resulting transactions remain admissible.

Transaction level. Authorization policies (E1–E3) constrain individual transactions (e.g., amount, recipient, rules) [53, 57]. However, they do not capture the execution history, cumulative spend, or multi-step strategies. Therefore, sequences of valid transactions may violate intended spending boundaries through repetition, fragmentation, or timing manipulation, and this effect is further

amplified by strategic agent behavior [81]. This limitation may extend to the multi-entity setting. Considering multiple colluding agents, they distribute actions across identities while remaining locally compliant, and reputation or stake mechanisms provide signals but do not enforce correctness of composed interactions [82].

Legal and institutional level. The delegated authorization does not ensure continued alignment with user consent or institutional expectations. Standing delegation is typically granted ex ante. However, the execution of subsequent payments lacks re-evaluation of purpose or cumulative impact. It creates a gap between formal permission and substantive consent. Thus, transactions may remain authorized, but actually exceed what users intended or would approve under evolving conditions.

Insight. Authorization acts as a local validity filter over individual transactions rather than a mechanism for preserving behavioral correctness over time, and cannot enforce consistency across sequences, delegation contexts, or institutional expectations. Thus, formally valid authorization may continue to legitimize spending even after its underlying trust assumptions have broken down.

5.3 Execution and Settlement Risks

Execution turns authorized intents into payment transactions. However, it treats payment success as completion, rather than the service completion in the real world. Once the payment is settled, the system moves forward even if the service is missing. In this sense, execution confirms that payment has been completed, without checking the promised result.

Technical level. Execution in blockchain systems is not instantaneous or deterministic, since transaction inclusion is subject to latency, reordering, and probabilistic confirmation [35]. Agents reacting to intermediate signals may observe delayed or inconsistent states, leading to retries or incorrect workflow progression. Execution also depends on auxiliary infrastructure (e.g., bundlers, relayers, paymasters) [55, 56]. It extends the trust boundary beyond the base protocol, where failures or manipulation may degrade reliability even if the underlying chain is secure.

Transaction level. The settlement of payment does not guarantee completion. There exist some off-chain architectures (S2) that improve throughput by deferring global settlement [23]. However, they introduce asynchronous, locally visible states that cannot be globally synchronized in time. Consequently, parties are left to interpret inconsistent signals, such as off-chain state updates versus on-chain inclusion. Thus, it is difficult to establish a single, authoritative finality point for transactions.

More fundamentally, payment completion is weakly coupled to service completion, since the execution only verifies a successful transfer, not correct delivery.

In inclusion-gated models such as x402 [53], completion is defined by transaction inclusion. In off-chain models, it is defined by the acceptance of intermediate states. In both cases, financial finality does not establish fulfillment, allowing providers to receive valid payment even if they do not deliver correct outcomes. There exist some mechanisms, such as A402 [26] and liquidity-aware or deferred-settlement protocols [60]. They aim for tighter coupling but are still confined to execution-layer coordination. Furthermore, they do not eliminate inconsistent interpretations of completion across workflows.

Legal and institutional level. In the sense of governance, execution does not determine the payment completion. Since settlement signals diverge from service outcomes, there is no shared anchor for different parties to assess fulfillment or liability. This gap would be amplified in agentic settings, because payment may be triggered automatically even if the service has not been delivered. Thus, execution can produce a valid settlement event but leaves economic and institutional completion unresolved.

Insight. Execution provides payment-completion signals rather than a shared notion of workflow completion, and does not ensure consistent observation, agreement on finality, or correct service delivery. Thus, payment finality can be operationally valid yet semantically incomplete.

5.4 Accountability and Privacy Risks

Accounting records how payments, outcomes, and responsible actors relate to each other, but the linkage is often incomplete and hard to verify.

Technical level. Blockchain serves as an immutable ledger to record transactions, but it cannot capture the off-chain execution details or decision processes. Transactions are tied to cryptographic identities, yet the causal chain, including interactions, external inputs, model reasoning and planning, is not recorded in a verifiable form [8, 83]. Thus, accounting establishs that a transfer occurred, but not why it occurred or which component in the user–agent–service chain was responsible, leaving causal attribution under-specified.

Transaction level. Accounting should evaluate whether payments correspond to successful service fulfillment. However, current systems only loosely link on-chain payment evidence (V1) with off-chain outcomes via logs or receipts. For example, x402-style flows prove payment occurrence [53], but their service outcomes are recorded separately. It only yields a *post hoc* correlation rather than unified binding. Consequently, valid payments may lack verifiable evidence of successful service delivery, leading to ambiguity in audit and dispute resolution. The core issue is the absence of an effective binding mechanism that can make payment and outcome jointly accountable.

Privacy and institutional level. Records that support accountability may also reveal sensitive personal information. Transparent transaction histories expose interaction patterns, counterparties, and timing, which may allow others

to infer workflows or decision strategies [84, 30]. This creates a basic trade-off. Greater transparency improves verification and auditing, but causes information leakage. Stronger privacy reduces exposure, but provides less evidence for accountability.

Insight. Accounting provides reliable payment records. But it does not guarantee the linkage among payment, outcome, and responsibility. Since evidence (V), coupling (C), and attribution remain only partially aligned, existing systems can only reconstruct causality post hoc, leading to a trade-off between incomplete attribution and excessive exposure. Thus, verifiable outcomes and records cannot balance accountability and privacy protection.

5.5 Authentication and Identity Risks

In existing agentic payments, authentication verifies access but does not establish identity or attribution. NIST distinguishes identity proofing, authentication, and federation as separate functions [85], a distinction that becomes more critical in agentic settings with delegated authority and cross-domain interactions [86]. Thus, a system may authenticate a key, account, or endpoint, yet remain blind to whether that action originates from the intended agent, aligns with a correct principal, or represents a stable identity across workflows.

In principle, meaningful identity should be able to handle the fields from protocol authentication to institution compliance. For example, FATF requires virtual-asset activity to support AML/CFT controls. So the counterparty attribution and screening are necessary [87]. Other systems, such as Coinbase KYT and Tracer, perform transaction screening, risk scoring, and entity attribution through address analysis and fund-flow tracing [88]. However, even when interactions are authenticated, identity may still be too weakly attributed to support admissibility or accountability.

Insight. Authentication verifies access but does not establish attribution, leaving systems unable to determine the responsible principal or assign accountability for authenticated actions.

5.6 Future Directions

Based on the above challenges, we identify a structural gap that agentic payment systems do not maintain consistency, control, and security across lifecycle stages. It points to three complementary directions.

Consistency: ensuring that payments correspond to actual outcomes. In agentic settings, consistency needs protocol-level support through a stronger *payment–service binding*. One promising direction is to use a shared append-only execution record that persists across the full lifecycle. It achieves commitment

of intent during discovery, adds policy decisions to the authorization, attaches settlement references at execution, and records the outcome evidence when accounting. As a result, all stages refer to the same anchored record. It enables the end-to-end consistency checking via a single execution trace.

Control: governing behavior, not just individual transactions. Future systems should not judge each payment in isolation. Instead, they should take past behavior into account, such as how much an agent has spent, how often it interacts, and with whom it interacts. They should also be able to tighten restrictions or revoke permissions when behavior starts to change. Incorporating a decentralized identity management system into it is a promising solution direction.

Composition: coordinating payments across workflows. Agentic payment can be regarded as a composition of interdependent actions across agents, systems, and execution contexts. For example, in multi-hop workflows, delegated tasks propagate payment obligations. It requires the dependency-aware consistency across many chained and concurrent interactions. This also spans heterogeneous execution environments, where different parts of a workflow execute across mixed settlement rails, such as on-chain settlement and off-chain service provisioning. It makes execution and accounting inherently cross-system.

Formation: negotiating payable terms across interactions. We observe that payment terms may not always be fixed upfront. For example, agent-mediated interactions can involve iterative or multi-round negotiation over price, volume, or service scope, as reflected in evolving x402-style pricing models such as "up-to" pricing and negotiated schemes [89, 90]. Therefore, lifecycle models should account for how payable conditions are formed across interactions, rather than assuming they are fully specified prior to authorization and execution.

6 Conclusion

In summary, while blockchain enables agents to make payments, it does not inherently guarantee their correctness. We systematize this space through a four-stage lifecycle decomposition model and show that existing designs provide partial guarantees at individual stages, failing to preserve correctness across the end-to-end workflow. The key gaps lie in intent binding, delegated control over evolving agent behavior, and the accountable linkage between payment and service outcomes. By identifying these limitations and outlining future research directions, we aim to contribute to the development of a secure and reliable A2A payment ecosystem.

Acknowledgment

This research is supported by the National Research Foundation, Singapore and Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in

this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore and Infocomm Media Development Authority.

References

1. Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.
2. OpenClaw Team. Openclaw. `https://openclaw.ai/`.
3. Shiping Chen, Qin Wang, Guangsheng Yu, Xu Wang, and Liming Zhu. Clawed and dangerous: Can we trust open agentic systems. *arXiv preprint arXiv:2603.26221*, 2026.
4. Yukun Jiang, Yage Zhang, Xinyue Shen, Michael Backes, and Yang Zhang. " humans welcome to observe": A first look at the agent social network moltbook. *arXiv preprint arXiv:2602.10127*, 2026.
5. Md Asadul Islam, Subbulakshmi Somu, and Faraj Mazyed Faraj Aldaihani. The rise of agentic ai: Synthesis of current knowledge and future research agenda. *Global Business and Organizational Excellence*, 45(3):402–416, 2026.
6. Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, et al. SoK: Agentic skills–beyond tool use in LLM agents. *arXiv preprint arXiv:2602.20867*, 2026.
7. Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
8. Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.
9. Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. Agentgen: Enhancing planning abilities for large language model based agent via environment and task generation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2025.
10. Mohamad Abou Ali, Fadi Dornaika, and Jinan Charafeddine. Agentic ai: a comprehensive survey of architectures, applications, and future directions. *Artificial Intelligence Review*, 59(1):11, 2025.
11. Aske Plaat, Max van Duijn, Niki Van Stein, Mike Preuss, Peter van der Putten, and Kees Joost Batenburg. Agentic large language models, a survey. *Journal of Artificial Intelligence Research*, 84, 2025.
12. Guangsheng Yu, Qin Wang, Rui Lang, Shuai Su, and Xu Wang. Plantwin: Privacy-preserving planning abstractions for cloud-assisted llm agents. *arXiv preprint arXiv:2603.18377*, 2026.
13. Partha Pratim Ray. A review on agent-to-agent protocol: Concept, state-of-the-art, challenges and future directions. *Authorea Preprints*, 2025.
14. Wenxin Xu, Taotao Wang, Yihan Xia, Shengli Zhang, and Soung Chang Liew. Agent-osi: A layered protocol stack toward a decentralized internet of agents. *arXiv preprint arXiv:2602.13795*, 2026.
15. David M Rothschild, Markus Mobius, Jake M Hofman, Eleanor W Dillon, Daniel G Goldstein, Nicole Immorlica, Sonia Jaffe, Brendan Lucier, Aleksandrs Slivkins, and Matthew Vogel. The agentic economy. *arXiv preprint arXiv:2505.15799*, 2025.

16. Clement Conand. Financial autonomous ai agents: Buying services and data without human intervention. *Available at SSRN 5907404*, 2025.
17. Diana Hancock and David B Humphrey. Payment transactions, instruments, and systems: A survey. *Journal of Banking & Finance*, 21(11-12):1573–1624, 1997.
18. Mastercard. How credit or debit card payment processing works, 2025. Mastercard merchant documentation.
19. World Wide Web Consortium (W3C). Payment request api, 2026. W3C Candidate Recommendation Draft.
20. Mastercard agent pay. `https://developer.mastercard.com/mastercard-checkout-solutions/documentation/use-cases/agent-pay/`.
21. Awid Vaziry, Sandro Rodriguez Garzon, and Axel Küpper. Towards multi-agent economies: Enhancing the A2A protocol with ledger-anchored identities and x402 micropayments for ai agents. *arXiv preprint arXiv:2507.19550*, 2025.
22. David GW Birch and Debbie Gamble. Agentic commerce and payments: Exploring the implications of robots paying robots. *Journal of Payments Strategy & Systems*, 19(1):72–84, 2025.
23. Kite AI Team. From human-centric to agent-native: Building trustless payment infrastructure for agentic ai, 2025. Whitepaper.
24. Yuexin Xiang, Yuchen Lei, Yuanzhe Zhang, Qin Wang, Tsz Hon Yuen, Andreas Deppeler, and Jiangshan Yu. Leveraging large language models to bridge cross-domain transparency in stablecoins. *arXiv preprint arXiv:2512.02418*, 2025.
25. Yuchen Lei, Yuexin Xiang, Qin Wang, Rafael Dowsley, Tsz Hon Yuen, Kim-Kwang Raymond Choo, and Jiangshan Yu. Large language models for cryptocurrency transaction analysis: a bitcoin case study. *arXiv preprint arXiv:2501.18158*, 2025.
26. Yue Li, Lei Wang, Kaixuan Wang, Zhiqiang Yang, Ke Wang, Zhi Guan, and Jianbo Gao. A402: Bridging web 3.0 payments and web 2.0 services with atomic service channels. *arXiv preprint arXiv:2603.01179*, 2026.
27. Vivek Acharya. Secure autonomous agent payments: Verifying authenticity and intent in a trustless environment. *arXiv preprint arXiv:2511.15712*, 2025.
28. Yuquan Li, Yuexin Xiang, Qin Wang, Tsz Hon Yuen, Andreas Deppeler, and Jiangshan Yu. Sok: Stablecoins in retail payments. *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2026.
29. Chong Chen, Jianzhong Su, Jiachi Chen, Yanlin Wang, Tingting Bi, Jianxing Yu, Yanli Wang, Xingwei Lin, Ting Chen, and Zibin Zheng. When chatgpt meets smart contract vulnerability detection: How far are we? *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 34(4):1–30, 2025.
30. Zehang Deng, Yongjian Guo, Changzhou Han, Wanlun Ma, Junwu Xiong, Sheng Wen, and Yang Xiang. Ai agents under threat: A survey of key security challenges and future pathways. *ACM Computing Surveys (CSUR)*, 57(7):1–36, 2025.
31. Bill Marino and Ari Juels. Giving ai agents access to cryptocurrency and smart contracts creates new vectors of ai harm. *arXiv preprint arXiv:2507.08249*, 2025.
32. Tianhao Li, Chuangxin Chu, Yujia Zheng, Bohan Zhang, Neil Zhenqiang Gong, and Chaowei Xiao. A2asecbench: A protocol-aware security benchmark for agent-to-agent multi-agent systems. In *International Conference on Learning Representations (ICLR)*.
33. Tanusree Debi and Wentian Zhu. Whispers of wealth: Red-teaming google's agent payments protocol via prompt injection. *arXiv preprint arXiv:2601.22569*, 2026.
34. Shenzhe Zhu, Jiao Sun, Yi Nian, Tobin South, Alex Pentland, and Jiaxin Pei. The automated but risky game: Modeling agent-to-agent negotiations and trans-

actions in consumer markets. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025.
35. Molud Esmaili and Ken Christensen. Performance modeling of public permissionless blockchains: A survey. *ACM Computing Surveys (CSUR)*, 57(7):1–35, 2025.
36. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
37. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837, 2022.
38. Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.
39. Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.
40. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.
41. Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform, 2014.
42. Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22. Association for Computing Machinery, 2023.
43. Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2024.
44. Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.
45. Stefanos Chaliasos, Denis Firsov, and Benjamin Livshits. Towards a formal foundation for blockchain zk rollups. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 2714–2728, 2025.
46. Yuanzhe Zhang, Shirui Pan, and Jiangshan Yu. Txallo: Dynamic transaction allocation in sharded blockchain systems. In *IEEE International Conference on Data Engineering (ICDE)*, pages 721–733, 2023.
47. Yuanzhe Zhang, Shirui Pan, and Jiangshan Yu. Mosaic: Client-driven account allocation framework in sharded blockchains. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 604–614, 2025.
48. Yuquan Li, Yuexin Xiang, Qin Wang, Tsz Hon Yuen, Andreas Deppeler, and Jiangshan Yu. Sok: Stablecoins in retail payments. *arXiv preprint arXiv:2601.00196*, 2026.
49. Agentcard. `https://a2acn.com/en/docs/concepts/agentcard/`.
50. Ethereum Improvement Proposals. EIP-8004: Agent Reputation and Trust Framework. `https://eips.ethereum.org/EIPS/eip-8004`, 2025.
51. Skyfire. `https://skyfire.xyz/`.

52. David Shi and Kevin Joo. Sybil-resistant service discovery for agent economies. *arXiv preprint arXiv:2510.27554*, 2025.
53. x402: An open standard for internet-native payments. `https://www.x402.org/x402-whitepaper.pdf`.
54. Machine Payments Protocol (MPP). `https://docs.stripe.com/payments/machine/mpp`.
55. Erc-4337 account abstraction documentation. `https://docs.erc4337.io/index.html`.
56. Ivan Dimitrov and Wolfgang Prinz. Leveraging AI agents for task automation in blockchain wallets with account abstraction. In *International Conference on Blockchain Computing and Applications (BCCA)*, pages 8–15. IEEE, 2025.
57. Jan Gorzny, Fatemeh Heidari Soureshjani, and Martin Derka. Account abstraction for enforcing blockchain-based ai agent non-functional requirements. In *2025 IEEE 33rd International Requirements Engineering Conference Workshops (REW)*, pages 359–364. IEEE, 2025.
58. Circle nanopayments documentation. `https://www.circle.com/nanopayments#features`.
59. Tempo. `https://tempo.xyz/`.
60. Jeshwanth Ravi. Agentic payments: The just-in-time liquidity protocol and the future of value exchange. *Sch J Eng Tech*, 3:137–142, 2026.
61. Ailiya Borjigin, Cong He, Charles CC Lee, Wei Zhou, and Cheng Yao Song. Ai agent architecture for decentralized trading of alternative assets. In *IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, pages 1–8, 2025.
62. ERC-2612: Permit Extension for EIP-20 Signed Approvals. `https://eips.ethereum.org/EIPS/eip-2612`.
63. Qin Wang and Shiping Chen. Account abstraction, analysed. In *IEEE International Conference on Blockchain (Blockchain)*, pages 323–331, 2023.
64. Permit2 Overview. `https://docs.uniswap.org/contracts/permit2/overview`.
65. EIP-2612 Gas Sponsoring Extension. `https://docs.x402.org/extensions/eip2612-gas-sponsoring`.
66. Saad Alqithami. Autonomous agents on blockchains: Standards, execution models, and trust boundaries. *arXiv preprint arXiv:2601.04583*, 2026.
67. Botao'Amber' Hu and Bangdao Chen. Insured agents: A decentralized trust insurance mechanism for agentic economy. *arXiv preprint arXiv:2512.08737*, 2025.
68. Andrea Muttoni and Jason Zhao. Agent TCP/IP: An agent-to-agent transaction system. *arXiv preprint arXiv:2501.06243*, 2025.
69. Vineeth Sai Narajala, Ken Huang, and Idan Habler. Securing genai multi-agent systems against tool squatting: A zero trust registry-based approach. *arXiv preprint arXiv:2504.19951*, 2025.
70. Mohamed Amine Ferrag, Norbert Tihanyi, Djallel Hamouda, Leandros Maglaras, Abderrahmane Lakas, and Merouane Debbah. From prompt injections to protocol exploits: Threats in llm-powered ai agents workflows. *arXiv preprint arXiv:2506.23260*, 2025.
71. Zibin Lin, Shengli Zhang, Guofu Liao, Dacheng Tao, and Taotao Wang. Binding agent id: Unleashing the power of ai agents with accountability and credibility. *arXiv preprint arXiv:2512.17538*, 2025.
72. Yedidel Louck, Ariel Stulman, and Amit Dvir. Security analysis of agentic ai communication protocols: A comparative evaluation. *arXiv preprint arXiv:2511.03841*, 2025.

73. Gillian K Hadfield and Andrew Koh. An economy of AI agents. *arXiv preprint arXiv:2509.01063*, 2025.
74. Tomer Jordi Chaffer. Can we govern the agent-to-agent economy? *arXiv preprint arXiv:2501.16606*, 2025.
75. Botao'Amber' Hu and Helena Rong. Inter-agent trust models: A comparative study of brief, claim, proof, stake, reputation and constraint in agentic web protocol design-a2a, ap2, erc-8004, and beyond. *arXiv preprint arXiv:2511.03434*, 2025.
76. Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *33rd USENIX Security Symposium (USENIX Sec)*, pages 1831–1847, 2024.
77. Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yuan Zhang, Michael Backes, Yun Shen, and Yang Zhang. Instruction backdoor attacks against customized {LLMs}. In *33rd USENIX Security Symposium (USENIX Sec)*, pages 1849–1866, 2024.
78. Qianlong Lan, Anuj Kaul, Shaun Jones, and Stephanie Westrum. Silent egress: When implicit prompt injection makes llm agents leak without a trace. *arXiv preprint arXiv:2602.22450*, 2026.
79. Tong Liu, Zizhuang Deng, Guozhu Meng, Yuekang Li, and Kai Chen. Demystifying rce vulnerabilities in llm-integrated apps. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 1716–1730, New York, NY, USA, 2024. Association for Computing Machinery.
80. Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM workshop on artificial intelligence and security*, pages 79–90, 2023.
81. Yunbo Long, Yuhan Liu, Liming Xu, and Alexandra Brintrup. Emodebt: Bayesian-optimized emotional intelligence for strategic agent-to-agent debt recovery. *arXiv preprint arXiv:2503.21080*, 2025.
82. David Joseph. Agency protocol: A decentralized trust-building system using domain-specific merit and economic stakes to incentivize promise-keeping in agent-to-agent interactions. 2025.
83. Md Monjurul Karim, Dong Hoang Van, Sangeen Khan, Qiang Qu, and Yaroslav Kholodov. Ai agents meet blockchain: A survey on secure and scalable collaboration for multi-agents. *Future Internet*, 17(2):57, 2025.
84. Muhammad Nasir Mumtaz Bhutta, Amir A Khwaja, Adnan Nadeem, Hafiz Farooq Ahmad, Muhammad Khurram Khan, Moataz A Hanif, Houbing Song, Majed Alshamari, and Yue Cao. A survey on blockchain technology: Evolution, architecture and security. *Ieee Access*, 9:61048–61073, 2021.
85. National Institute of Standards and Technology. Digital identity guidelines. NIST Special Publication 800-63, Revision 4.
86. Tobin South, Subramanya Nagabhushanaradhya, Ayesha Dissanayaka, Sarah Cecchetti, George Fletcher, Victor Lu, Aldo Pietropaolo, Dean H Saxe, Jeff Lombardo, Abhishek Maligehalli Shivalingaiah, et al. Identity management for agentic ai: The new frontier of authorization, authentication, and security for an ai agent world. *arXiv preprint arXiv:2510.25819*, 2025.
87. Financial Action Task Force. Virtual assets: Targeted update on implementation of the fatf standards on vas and vasps. https://www.fatf-gafi.org/en/publications/Fatfrecommendations/targeted-update-virtual-assets-vasps-2024.html.
88. Coinbase. Crypto compliance at scale: Coinbase tracer and know your transaction api. https://www.coinbase.com/blog/introducing-coinbase-intelligence-crypto-compliance-at-scale.

89. x402. FAQ. https://docs.x402.org/faq, 2026.
90. mjohnson518. feat: Add negotiated payment scheme for dynamic pricing. https://github.com/coinbase/x402/pull/628. GitHub Pull Request #628, coinbase/x402.