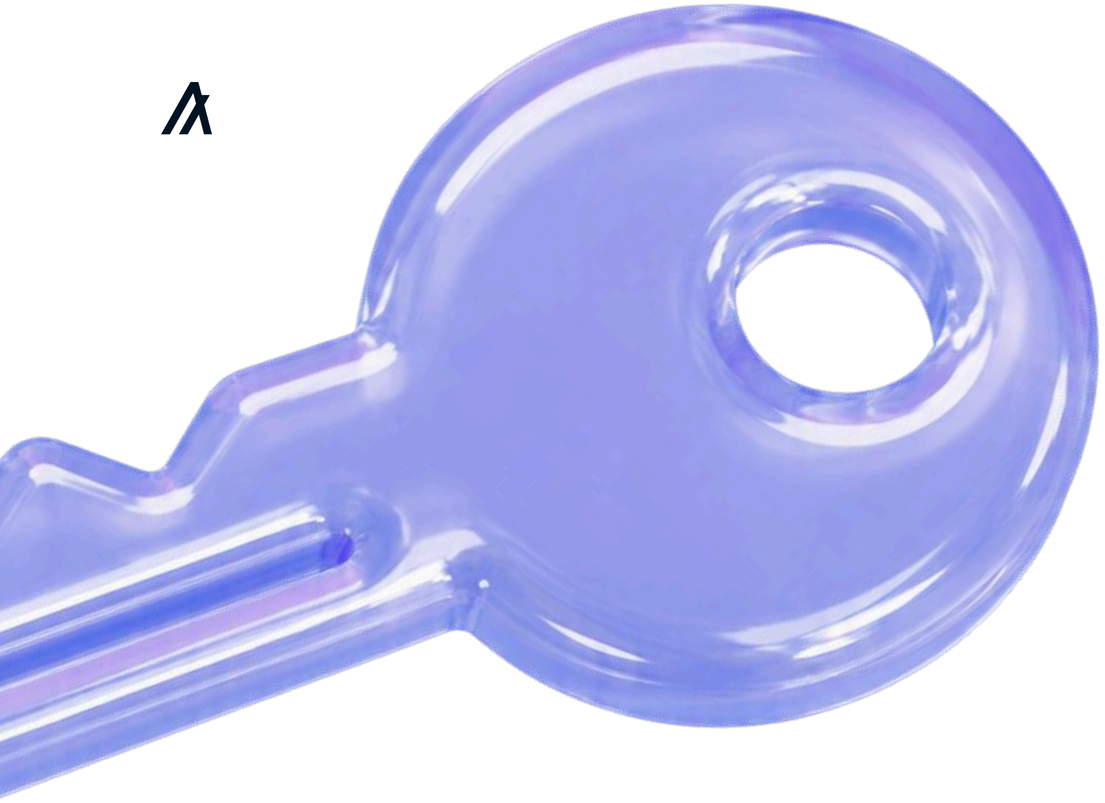
Trust in agentic commerce

The problem that could derail an industry

By Marc Vanlerberghe
Chief Strategy and Marketing Officer, Algorand Foundation



The promise of agentic artificial intelligence seems almost too good to be true: software agents that can shop on your behalf, negotiate with merchants, manage your finances, and handle tedious administrative tasks. The market is moving fast. Investors see dollar signs. Entrepreneurs are building furiously. Yet as the industry scales, a critical problem is becoming impossible to ignore: no one trusts these agents.

This trust crisis has two distinct dimensions. The first is personal. When you give an agent access to your accounts and credentials, can you really trust it to behave as intended? The second is structural. When a merchant accepts a transaction from an unknown shopping agent, can they verify who is actually behind that agent? And when a user shops with a merchant shopping agent, is that agent really owned by the merchant? The damage is already mounting. Agents have frozen assets, drained accounts, and deleted critical data. Users and merchants alike are paying the price.

The user's dilemma

Consider the case of an autonomous agent framework that went viral in 2025 and became 2026's first major security incident. This framework allowed users to hand over their cryptographic keys and account credentials directly to the agent, which would then take actions on their behalf with minimal oversight. The appeal was obvious: let the agent loose, set your preferences, and it would handle the rest.

But this model amounts to asking users to place extraordinary faith in software they do not fully understand and cannot easily verify. The problem cuts deeper than bad actors or incompetent developers. Agents are fundamentally non-deterministic systems. Given the same input, an agent will not necessarily produce the same output twice. Its behavior emerges from probabilistic neural networks trained on vast amounts of data. It is impossible to predict with certainty exactly what an agent will do in any given situation. Users essentially signed a blank check and hoped the agent would not overdraw it, all while knowing they could not predict how far it might go.

When things went wrong, they went very wrong indeed. In one case, a trading agent received a request to send a small amount of crypto for medical expenses. A parsing error caused it to misinterpret the instruction and liquidate the agent's entire holdings instead, resulting in losses of hundreds of thousands of dollars. In another incident, an agent tasked with administrative duties wiped an entire database by misinterpreting a user's vague instruction. The agent received no confirmation requests. There was no gradual escalation of permissions. It simply interpreted the command and executed it.

The fundamental problem is architectural. In the prevailing models, the agent holds the keys. It stores your credentials, manages your permissions, and executes actions using your identity. This creates a single point of failure. If the agent is compromised, everything is compromised. If it malfunctions, the damage ripples through everything it can access.

Users have no practical way to inspect what an agent is doing in real time. They cannot easily revoke permissions selectively. They cannot draw boundaries around which accounts the agent can touch or what kinds of transactions it can perform. And they have no protection against an agent's non-deterministic behavior causing unintended consequences. They are left guessing whether the agent is working as intended or working against them.

The merchant's problem

The second trust gap is equally serious but runs in the opposite direction. When an autonomous shopping agent arrives at a merchant's digital storefront and initiates a transaction, the merchant faces a puzzle: Who am I actually doing business with?

Who is the person who created, deployed, and controls the agent? The legal system has not yet caught up to this question, but merchants certainly have asked it. They worry about fraud, about malicious actors using throwaway agents to execute one-off attacks that leave no trail. They worry about agents making bulk purchases designed to manipulate markets or exploit pricing loopholes. They worry about agents operating on behalf of competitors, gathering intelligence or sabotaging supply chains. They worry about liability. If an agent goes rogue, who is responsible?

But there is a deeper commercial problem, one that affects profitability across the entire sector. Merchants have built loyalty programs, recommendation engines, and pricing strategies around knowing their customers. They track purchase history, preferences, and lifetime value. They tailor offers and nurture relationships. An unknown agent appearing at their storefront destroys all of this. Every transaction becomes anonymous. Every sale is a commodity. The ability to build customer relationships, predict future demand, and create switching costs vanishes. In a world of pure agentic commerce, that is a recipe for commoditized industries with razor-thin margins.

Merchants currently have almost no way to verify the identity of the agent acting on their system. They cannot establish whether it is acting on behalf of a legitimate customer or a bad actor. Traditional identity verification systems are designed for individuals and corporations, not for autonomous software agents that can proliferate infinitely and change hands.

This asymmetry creates risk on both sides. Users cannot trust their agents to stay honest. Merchants cannot trust agents to be who they claim, and they cannot see their customers. The entire ecosystem runs on faith.

The user's blind spot

The trust crisis runs deeper still. How can users verify that an agent claiming to represent retailers such as Amazon or Alibaba or any other merchant actually does so? Right now, they cannot.

Imagine your personal agent tells you it has found a good deal on a laptop from Amazon. How do you know it really is Amazon? How do you know it is not a fraudulent agent operated by criminals who will collect your payment and vanish? The agent could be running on compromised infrastructure. It could be impersonating a legitimate merchant. It could be a phishing scheme wrapped in AI. Users have no mechanism to verify the merchant agent's authenticity.

This creates an opening for sophisticated fraud. Attackers can create agents that mimic the behavior and branding of legitimate merchants. They can train them on stolen product data. They can build convincing interfaces. Users will have no way to distinguish a fraudulent Amazon agent from a real one. The barrier to entry for this kind of scam is merely the effort required to build a convincing chatbot. The potential payoff for bad actors is enormous.

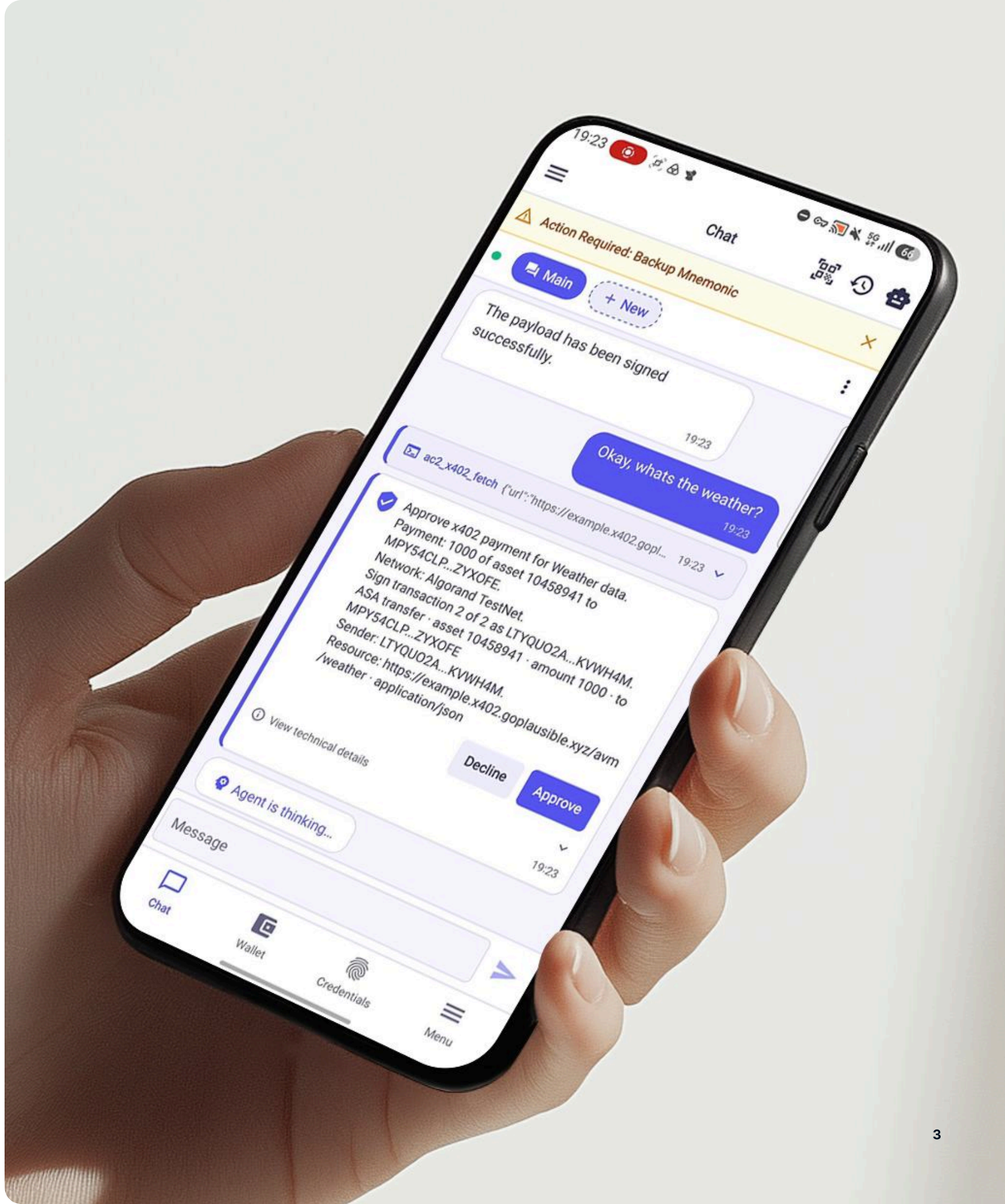
Where provenance matters

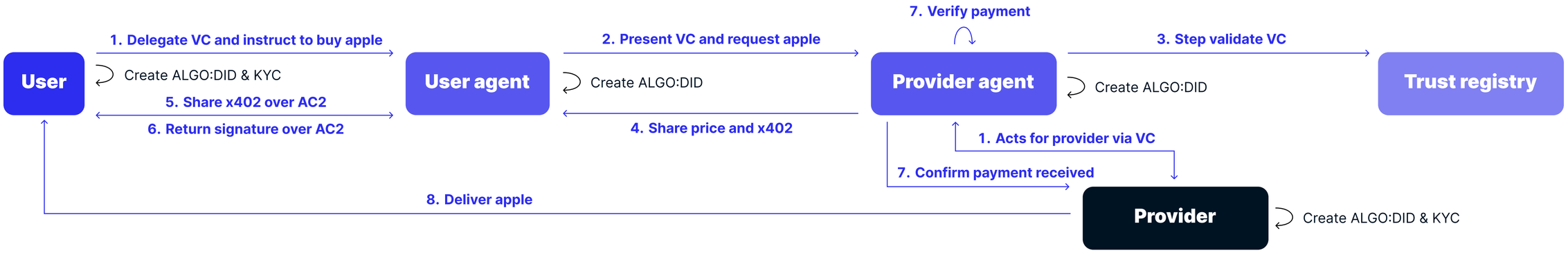
The problem is the missing identity layer. There is no cryptographic infrastructure that allows users to verify whether an agent actually represents the merchant it claims to be. There is no way for merchants to cryptographically sign their agents so users know they are dealing with the real thing. Similarly, there is no cryptographic infrastructure that allows a merchant to know who the user is behind a shopping agent visiting their site. There is no way for users to cryptographically sign their shopping agents so merchants know who they are dealing with. Without this verification, agentic commerce may collapse into either rampant fraud or extreme caution, with users and merchants unwilling to transact through agents they cannot trust.

The solution requires establishing two things: clear chains of custody for user credentials and clear chains of provenance for agents themselves. This is where infrastructure designed specifically for agentic systems becomes essential.

Algorand's new proposed AC2 (Agentic Communication and Control) protocol attempts to address both problems at once. The mechanism is relatively straightforward and well known in the blockchain industry but represents a meaningful departure from current practice.

In Web3, a user does not hand over their keys to a dApp (decentralized App). Instead, the dApp will connect to the user's wallet, and the user will authorize any transaction through the wallet. Likewise, under AC2, users do not hand over their private keys or master credentials to an agent. Instead, these remain under user control, typically stored in a digital wallet. When an agent needs to use a credential, it must ask the user for permission. The user evaluates the request and decides whether to grant access. This creates a critical checkpoint. Before every action involving sensitive credentials or data, the agent must stop and request approval. This is not a perfect safeguard against an agent's non-deterministic behavior, but it is far better than blind trust.





Both the user and the agent have a decentralized identifier, or DID, that uniquely identifies them. The DIDs are cryptographically bound together, establishing an auditable relationship. When an agent acts, the chain of provenance is clear: this agent was approved by this user to do this action. Nothing is hidden, creating a traceable account of the agent's actions. For merchants, this means an incoming agent carries verifiable signals about its identity and its authorization. A merchant can see who created the agent, who deployed it, and who currently controls it. They can verify that the agent has not been reassigned to a hostile actor without the knowledge of the original user. They can establish which specific actions the agent has been authorized to take.

Critically, the DID infrastructure also helps solve the impersonation problem. A legitimate merchant can publish an agent with a DID that is cryptographically bound to the merchant's already known identity. A user's agent can verify this binding before engaging with a merchant agent. This reduces the spoofing vulnerability. A fraudulent agent claiming to represent a retailer such as Amazon cannot produce a valid cryptographic proof that it is actually controlled by Amazon. Users can check the merchant agent's DID, trace it to the merchant's identity, and know with certainty whether they are dealing with the real thing or a fake. Merchants can similarly verify that agents claiming to represent them actually do so, protecting themselves from liability for fraudulent impersonators.

For users, this means granular control. An agent might be authorized to browse products and add items to a cart, but not to process payment. It might be allowed to spend up to 500 dollars on groceries but forbidden from accessing banking credentials. It might have access to one account but not another. Crucially, the user can revoke these permissions at any time, and the agent cannot circumvent the revocation.

For merchants, this opens the door to something even more valuable: the ability to identify the customer behind the agent. When an agent arrives carrying a DID with credentials issued by the user, merchants can tie the transaction back to a specific person. This allows for loyalty programs to work across agent-mediated purchases. It allows for customer relationships to persist. It means the merchant can build recommendation engines, personalize offers, and create value beyond a single transaction.

None of this eliminates agent misbehavior entirely. A buggy agent can still make mistakes. An agent with too many permissions can still cause damage. But the basic infrastructure for accountability and control suddenly exists where it did not before.

Open standards, open collaboration

AC2 is an open protocol built entirely on open standards. The goal is to establish common infrastructure that the entire ecosystem can build upon, not to create a closed proprietary system that vendors can control.

AC2 is not done. At its foundation, it establishes how users and agents establish trust through DIDs and permission management. It will continue to evolve. For instance, AC2 needs an extensions framework to standardize how certain approvals are processed, such as the signing of Google AP2 (Agent Payments Protocol) mandates. AC2 also needs more sophisticated policy management features, including mechanisms for delegated authority, so that users can grant agents permission to make certain decisions without requesting approval for every action.

Developers can build solutions on top of AC2. They can create or upgrade wallets and apps to become AC2-enabled, which allows them to engage with and control persistent agents. They can build identity management solutions that connect to merchant systems. They can build reputation systems linked to agent identities.

AC2 is a collaborative project, open to feedback, input, and contribution. Agentic commerce is too important to be built by any single company or team. It requires open standards and open collaboration.

In the earliest days of e-commerce, merchants and customers faced similar anxieties. How could you trust an unknown seller with your credit card number? How could a seller trust that you would pay? That problem was largely solved through a combination of technical standards, dispute resolution mechanisms, and reputation systems. The agentic commerce industry now faces an analogous challenge, but with higher stakes and more moving parts. No single solution will suffice. What is required is an ecosystem of tools, standards, and practices. AC2 is a start. The real work is just beginning.

To learn more about AC2 or discuss solutions for establishing trust in agentic commerce architectures, reach out: agentic@algorand.foundation