You Are Your Keys (Sorry)

Wallet-as-Identity, Sealed Disclosure, and Capability-Grant Access for Agents

Lewis Tham
Unbrowse AI
lewis@unbrowse.ai

Cayden Chik
Unbrowse AI
cayden@unbrowse.ai

June 2026

Abstract

An agent that signs every layer it touches [1] has solved *how* an action is attested, but not the two questions that decide whether it should run at all: *who* is asking, and *who may see the answer*. This paper makes the narrow claim that those two questions collapse into one act — a signature — and need no account system, no bearer-token vault, and no per-resource ACL service to answer. Concretely: (i) a wallet signature is a first-class identity — the same Ed25519 key [3] that signs requests *is* the principal, with no web2 account required beneath it; (ii) every cached value is sealed to the wallet and revealed only under signature, fail-closed, so disclosure is a cryptographic consequence of identity rather than a policy check; and (iii) read access between principals is a signed, scoped, revocable capability grant in the object-capability tradition [4, 5], where the granter's signature is the sole authority and there is no ambient permission. Authentication, selective disclosure, and authorisation are three readings of one signed object. We report what ships as running code, and we are explicit about the standard guarantees this design does *not* provide (revocation latency, key loss, metadata exposure), because an identity paper that claims to have closed those is not to be trusted.

1 Introduction: who, and who may see

The companion security paper establishes that one Ed25519 key can sign every layer an agent descends through — screen, browser, CLI, OS, kernel, packet — so that no action crosses a boundary unsigned [1]. That is a statement about *integrity of action*. It leaves open two orthogonal questions that every multi-agent system must answer before an action is even admissible:

1. **Authentication** — who is this principal? Conventionally answered by an account record and a credential the principal presents.
2. **Authorisation and disclosure** — may this principal read this resource? Conventionally answered by an access-control list consulted at read time, and a secret store that hands out the bytes once the check passes.

The conventional answers each introduce a trusted third surface: an account database, an ACL service, a secret vault. Each is a place where authority is *asserted* rather than *proven*, and therefore a place an adversary can route around or a misconfiguration can leak. This paper's claim is that for an agent that already carries a signing key, all three surfaces disappear into the key itself. Identity is the public key; disclosure is decryption under the private key; authorisation is a signature by the resource owner naming who may decrypt. There is nothing left to consult.

This is not a new cryptographic primitive — it is a composition of well-understood ones (digital signatures, authenticated encryption, object capabilities) arranged so that the answer to "who, and who may see" is read off a signed object instead of fetched from a service. The contribution is the arrangement and the demonstration that it ships.

2 Related work

Wallet / key-as-identity. Public-key-as-principal is the oldest idea in the field: SPKI/SDSI binds authority to keys rather than names, and the object-capability literature makes the key the unforgeable handle [4]. Our contribution is not the idea but its operational shape for agents: the signing key the agent uses for action integrity is reused, unchanged, as the authentication principal, so there is no separate enrolment step and no account to phish.

Selective disclosure. Anonymous-credential systems let a holder reveal attributes without revealing more [6]; sealed-sender and end-to-end encryption bind plaintext to a key. We use the simplest instance of the family: authenticated encryption keyed by a value derived from the wallet, so a stored record is opaque to everyone but the holder, and opening it re-verifies the content-address. Disclosure is not a policy decision taken by a server; it is the ability to decrypt.

Capabilities over ACLs. The object-capability model replaces "check the subject against a list" with "hold an unforgeable, attenuable token" [4]; macaroons make such tokens delegatable and caveat-scoped with only a keyed MAC [5]. Our grants are the signed instance: a capability is a row signed by the granter, naming a grantee (or a lineage pattern), a scope, and an expiry; verification is signature-check plus scope-match, with no central authority in the loop.

Transparency. That every grant and every settled claim is appended to a hash-chained, append-only log is the Certificate-Transparency discipline [7] applied to authorisation: revocation and issuance are themselves logged events, so the access history is auditable rather than implicit in a mutable table.

3 Method: one signed object, three readings

3.1 Authentication — the signature is the principal

A request carries an Ed25519 signature over its canonical body. Verification recovers the public key; *that key is the identity.* There is no lookup that can fail open: an unsigned or mis-signed request has no principal and is rejected. A key may optionally be bound to a web2 account for convenience, but the binding is additive — absent it, the principal is simply `wallet:`⟨`pubkey`⟩, a fully usable standalone identity. **A wallet signature is a first-class identity**; the account, where present, is a label on top of it, never the root of it. *[shipped]*

3.2 Disclosure — sealed to the key, opened by the key

Every value the agent stores is sealed under authenticated encryption with a key derived from the wallet secret, and the stored record holds only a commitment (a content hash) plus opaque ciphertext. A value is **revealed only under signature, fail-closed**: presenting the wrong key

fails the authentication tag and returns nothing; presenting the right key decrypts and then re-checks the content-address before the plaintext is trusted. Disclosure is therefore a property of *holding the key*, not of passing a server-side permission check — there is no read endpoint that, misconfigured, hands the bytes to the wrong caller. *[shipped]*

3.3 Authorisation — a signed, scoped, revocable capability grant

For one principal to read another's data, the resource owner signs a **scoped, revocable capability grant**: a record naming the grantee (an exact key or a lineage pattern), the scope it authorises, an expiry, and the granter's signature as the sole authority. A read is admitted only if a grant exists whose signature verifies against the granter's key, whose scope matches the request, and which has not expired or been revoked; otherwise it fails closed. There is no ambient authority and no central policy server — the grant *is* the permission, and revocation is an appended event that the next verification observes. *[shipped]*

3.4 Why the three collapse into one

Each reading is the same object viewed from a different angle. The principal is a public key. Disclosure is decryption under its private half. Authorisation is a signature *by* a key naming another key. Authentication, selective disclosure, and access control are not three subsystems that must agree; they are three uses of the one signing identity, which is why none of them needs a trusted third surface to adjudicate.

4 Honest gaps — what this does not solve

An identity design that claims to have closed the hard problems is lying; we name the open ones as plainly as the wins.

Concern	Status	Honest reading
Revocation latency	open	a revoke is an appended event; a verifier that has not yet observed it can still ad
Key loss / recovery	open	losing the private key loses the identity and every sealed value under it
Metadata exposure	open	ciphertext hides values, not the existence/shape of the access graph
Grant explosion	partial	lineage-pattern grants bound the count, but broad patterns trade precision for co

Table 1: Open problems this design does not close. Each is a standard hard problem in capability systems; naming them is the point.

These are not incidental. Revocation in a log-based capability system is eventually-consistent by construction; key loss is the cost of having no account to reset against; metadata privacy needs machinery (mixing, padding) this design does not include. The contribution is the collapse of the three positive questions into one signed object, *not* a claim to have also solved revocation, recovery, and traffic analysis.

5 Conclusion

Once an agent carries a signing key for the integrity of its actions, that key is already enough to answer who is acting and who may see the result: identity is the public key, disclosure is decryption

under its private half, and authorisation is a signed, scoped, revocable grant naming another key. Authentication, selective disclosure, and access control collapse into three readings of one signed object, removing the account database, the ACL service, and the secret vault as trusted surfaces. We ship the three readings as running code and we decline to claim the hard residue — revocation latency, key loss, metadata exposure — is solved. The line between the collapse (real) and the residue (open) is the honest core of the result.

References

[1] L. Tham, C. Chik. *Crypto Was All You Needed: A Signed, Layer-Descending Stack for Agent Computation*. Unbrowse AI, 2026.

[2] L. Tham et al. *Internal APIs Are All You Need*. arXiv:2604.00694, 2026.

[3] S. Josefsson, I. Liusvaara. *Edwards-Curve Digital Signature Algorithm (EdDSA)*. RFC 8032, IETF, 2017.

[4] M. S. Miller. *Robust Composition: Towards a Unified Approach to Access Control and Concurrency Control*. PhD thesis, Johns Hopkins University, 2006.

[5] A. Birgisson, J. G. Politz, U. Erlingsson, A. Taly, M. Vrable, M. Lentczner. *Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud*. NDSS, 2014.

[6] J. Camenisch, A. Lysyanskaya. *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*. EUROCRYPT, 2001.

[7] B. Laurie, A. Langley, E. Kasper. *Certificate Transparency*. RFC 6962, IETF, 2013.