Cryptography for Distributed Systems, TUM

Trusting agents is a cryptography problem

Christian Pfrang · July 16, 2026

info@christianpfrang.com

Agenda

Part 1 · Agentic Commerce: The Landscape

What agents are, the payment stack, stablecoins, and the merchant-verification gap.

Part 2 · Keeping Secrets from Your Own Agent

Credential isolation, and use cases for zero-knowledge proofs

Part 3 · Compute as a commodity

Settlement, attestation, verification, paying on proof.

Part 1

Agentic Commerce: The Landscape

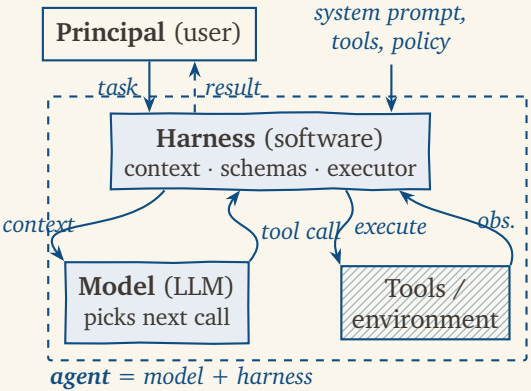
What is an agent?

An operational definition

An **agent** is an LLM pursuing a goal **on behalf of** a principal: it picks **tool calls**, a **harness** executes them — in a loop.

task: goal, not procedure · **context**: transcript = memory · **stop**: soft (model) / hard (budget)

```
context = [system_prompt, task]
while budget_ok():              # hard stop
    r = llm(context, tools)     # model decides
    if not r.tool_calls:        # soft stop
        return r.text           # -> principal
    context += [r, execute(r.tool_calls)]
```



Anthropic, Building effective agents (2024); Wooldridge & Jennings (1995)

Why agents find it hard to pay for things

Practical difficulties

- **Agents look like attack traffic** — exactly the bots that CAPTCHAs and WAFs exist to block.
- **Checkout is built for human eyes** — forms, pop-ups, dark patterns: each halts an autonomous flow.
- **Prompt injection** — a hostile product page can steer the model's choices.
- **Credential handling** — raw card data in an LLM harness is reckless (→ Part 2).

Regulatory difficulties

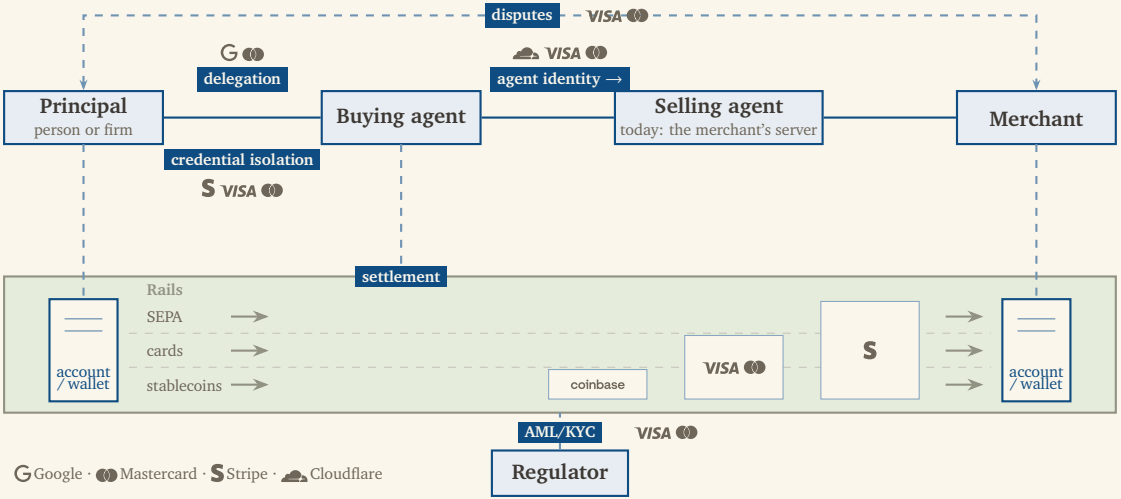
- **SCA under PSD2/PSD3** — the *payer* approves one amount, one payee; not an agent, later.
- **AML/KYC** — who transacts? Stablecoins add MiCA and the Travel Rule.
- **Contract formation & consent** — who consents to the agent's "I agree"?
- **AI Act & transparency** — agents must disclose that they are agents.
- **Consumer protection** — withdrawal rights assume a human reader.

Visa, Threats and risks of agentic commerce (2026); Taylor Wessing, Agentic AI in payments (2026)

What has to be solved — five sine qua nons

Agent identity	Tell an accountable agent from attack traffic — else the CAPTCHA wall. → a **keypair** per agent; identity = its public key, bound to nothing (wallet), by a registry entry (TAP), or by an issuer's signature (VI).
Delegation	A machine-checkable record of what the human authorized: how much, with whom, until when. → the customer **signs the agent's public key + limits**: a power of attorney as a signature chain.
Credential isolation	The agent spends *without holding* the card. → a **one-shot stand-in** — signed mandate (math enforces the cap) or vault token (a database does); a hijacked agent burns one token, never the card.
Settlement	Money moves at machine tempo and size. → a **signed transfer** some rail executes: on-chain (x402) or tokenized card rails.
Disputes & AML/KYC	Agents will err: someone must own reversal, liability, evidence. → the **signature chain is the evidence** — a non-repudiable audit trail.

Where the five live



Who solves what — four camps, five requirements

	Identity	Delegation	Cred. isolation	Settlement	Disp./AML/KYC	The gate
Stripe + OpenAI	OpenAI	Stripe	Stripe	Stripe	VISA · Mastercard	chat window
ACP · MPP	VISA · Mastercard	OpenAI		VISA		~4% + processing
Google alliance	Cloudflare	Google	VISA · Mastercard	Stripe	VISA · Mastercard	search & ads
UCP · AP2	VISA · Mastercard	Mastercard	Stripe	coinbase		Search · Gemini · YouTube
Card networks	VISA · Mastercard	VISA · Mastercard	VISA · Mastercard	VISA · Mastercard	VISA · Mastercard	the rail
tokens · TAP · VI	Cloudflare	Google				interchange + tokens
Coinbase + Cloudflare	Cloudflare	Google	—	coinbase	VISA	door & float
x402				Stripe		per-call tolls · reserves

OpenAI · Stripe · Google · Mastercard · Cloudflare

■ colour = core member · ■ grey = associated member

memberships as of 2026-07-10: cloudflare.com, ap2-protocol.org, developers.openai.com, mpp.dev

MPP AP2 TAP VI bind agents to a human — x402 doesn't

Binding: key ↔ human identity

The identifier is *human-legible*: a merchant, a domain, a cardholder — a claim about the world, so someone must attest it **externally**: issuer, CA, registry ⇒ classical PKI, delivered keys.

- **RSA** — the bulk of web certificates
- **ECDSA / P-256** ("ES256") — TLS, passkeys, SD-JWT mandates (VI, AP2)
- **EdDSA / Ed25519** — SSH, Web Bot Auth (TAP)

No binding: the key *is* the identity

Without binding, there is no key distribution. Therefore verification can not "ask" for an entity's public key.

Thankfully, this is exactly how Ethereum wallets are implemented (EIP-7120).

Message hash of public key & (secret-key) signed signature recovers the public key. I can use this to "accept" signature & wallet address.

Verify or recover the key

Binding: key ↔ human identity

RSA, Ed25519 (SSH, Signal), ES256 (= ECDSA/P-256, for VI TAP): one interface, keys **bound** via certificates and registries.

> KeyGen() → (sk, pk) — **bind & deliver** pk

VI TAP: $sk \xleftarrow{\$} \mathbb{Z}_n;\ pk = sk \cdot G$

> Sign$(m, sk) \to \sigma = (r, s)$

VI TAP: $h := \mathsf{H}(m)$; nonce $k \xleftarrow{\$} \mathbb{Z}_n$; $R = kG$; $r = R_x \bmod n$; $s = k^{-1}(h + r \cdot sk) \bmod n$

> Verify$(m, (r, s), pk) \to \{0, 1\}$

VI TAP: $h := \mathsf{H}(m)$; accept iff $\left[hs^{-1}G + rs^{-1}pk\right]_x \bmod n = r$

No binding: the key *is* the identity

Ethereum wallets: ECDSA on secp256k1. No binding ⇒ no PKI to serve pk ⇒ must recover.

> KeyGen() → (sk, pk) — **nothing to bind**

x402: $sk \xleftarrow{\$} \mathbb{Z}_n; pk = sk \cdot G$; addr:=H$(pk)$[12:32]

> Sign$(m, sk) \to \sigma = (r, s, v)$

x402: r, s as ES256; v: parity(R_y), recorded

> Verify$(m, (r, s, v), \text{addr}) \to \{0, 1\}$

x402: $h := \mathsf{H}(\texttt{0x 19 01} \,\|\, \Delta \,\|\, \mathsf{H}(m))$; rebuild R: $x = r$ ($+n$ if v's 2nd bit), $y = (x^3{+}7)^{(p+1)/4}$, sign by v's parity; $pk' := r^{-1}(sR - hG)$; accept iff H(pk')[12:32] = addr

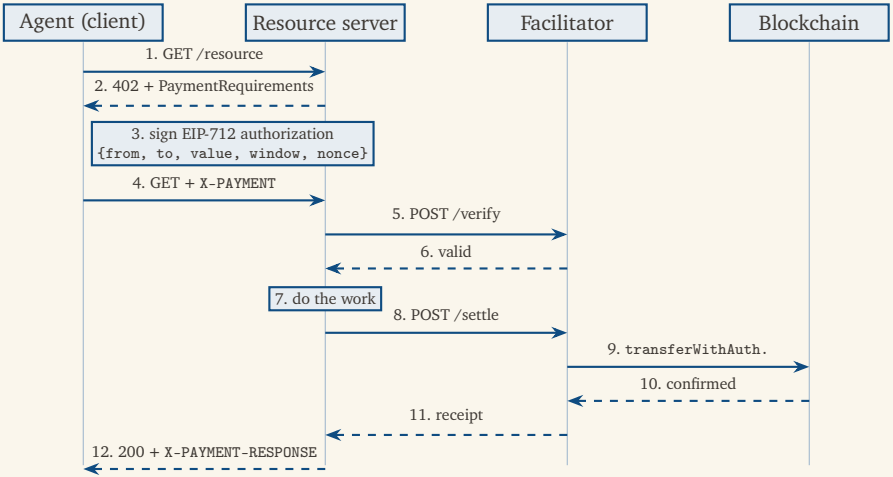
A curve point = (R_x, R_y) — two mirror points per x. H*: SHA-256 in ES256, Keccak-256 on Ethereum (h per EIP-712;* Δ *= chain + contract, anti-replay). RSA: no curve, just the group* $\mathbb{Z}_N^\times$*; Ed25519: R sent whole.*

Johnson–Menezes–Vanstone, ECDSA (Int. J. Inf. Sec. 2001); SEC 1 v2.0 §4.1.6; Ethereum Yellow Paper, App. F; EIP-712

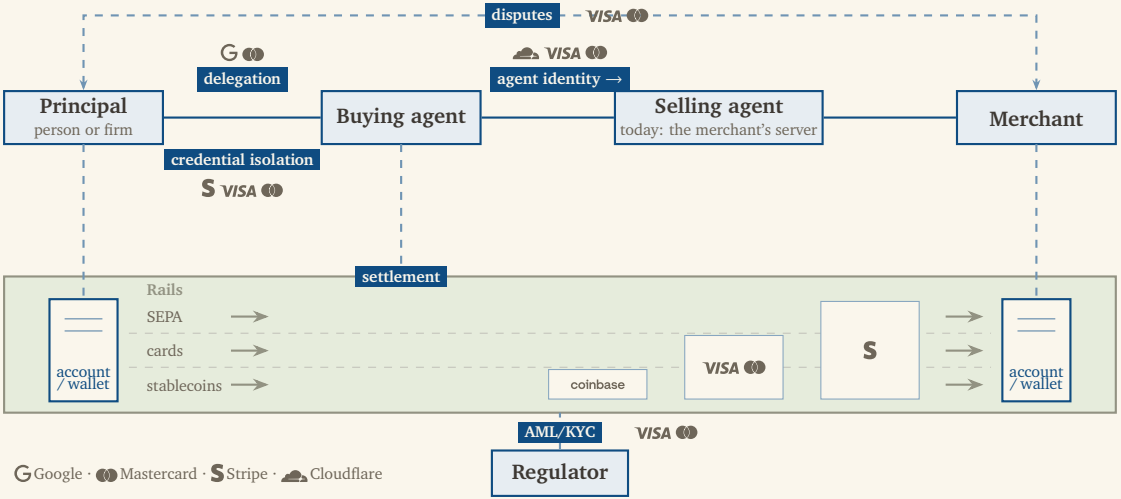
x402 — the exact protocol

Why x402?

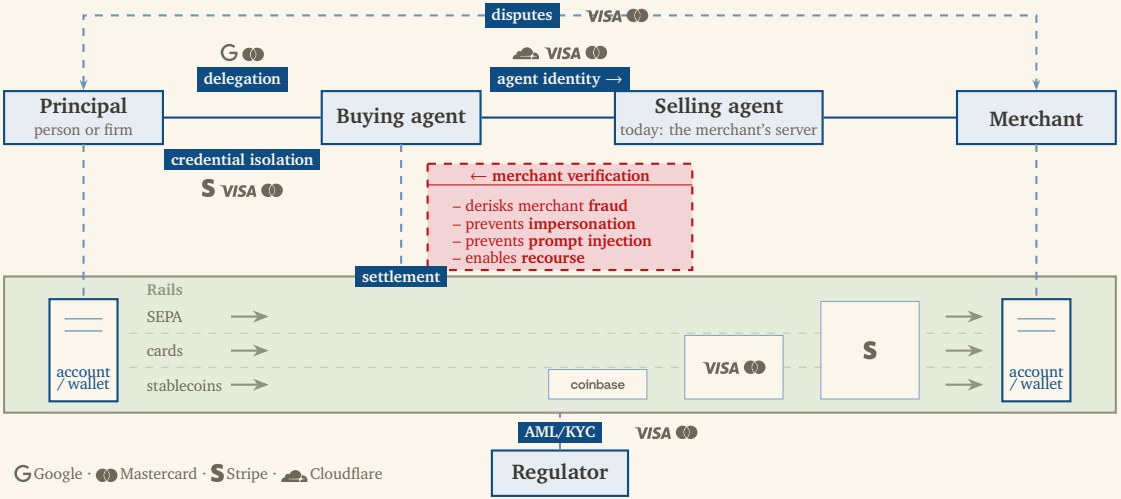
- no directory, no vault — *decentralized*
- agent identity = wallet
- open spec, no incumbent



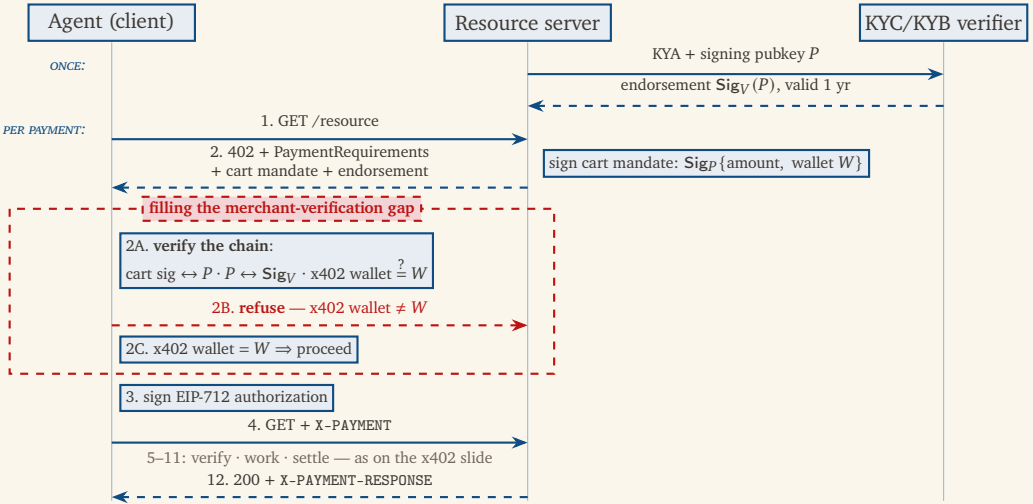
Something important is missing



Something important is missing: merchant verification



Closing the gap — a trust chain for the payee



Part 2

Keeping Secrets from Your Own Agent

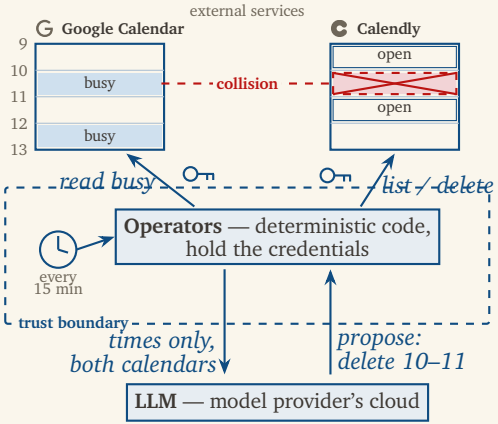
The calendar problem — your agent needs both your accounts

The task — and the constraint

- every 15 min, kicked off by a scheduler
- delete **Calendly** slots colliding with **Google**
- the **LLM** computes & proposes deletions
- model sees *times only* — nothing else

The threat model

- context **leaves the house**: provider logs
- **prompt injection**: event text is hostile
- **credential in context** = credential lost



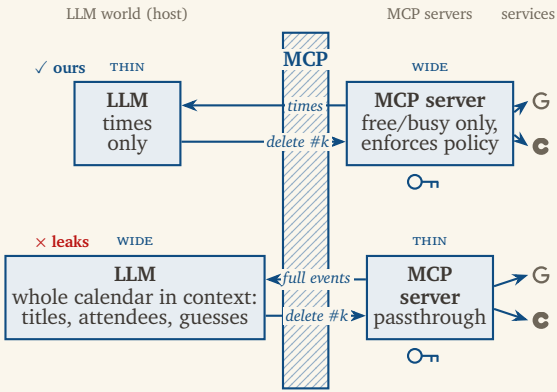
This shape has a name — MCP as a trust boundary

Model Context Protocol (MCP)

- **standardized** exactly this shape
- how LLM apps call **tools**, local or remote
- one standard instead of one integration per app–tool pair
- model sees **calls & results**, never tokens

From Anthropic to agentic commerce

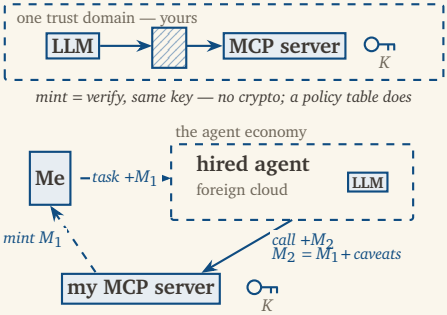
- **Anthropic** 2024 → Linux Fdn. 2025
- servers **charge per call**: **x402**



which computation runs on which side, and which data enters the LLM context, is a design choice

MCP spec; anthropic.com (2025-12-09); x402.org

Renting an agent — why the credential must shrink



Why OAuth doesn't work here

- **re-issued** per hop — authority *grows*
- narrowing: coarse scopes, **online-only**; **refresh** outlives expiry
- AS round-trips: **latency** · **reachability** (hop n unknown) · **metadata** (sees the graph)

Macaroons are the solution

- mint **once** — no AS in any loop
- every holder can **shrink** it: one HMAC
- verify with **one key** — authority only ↓

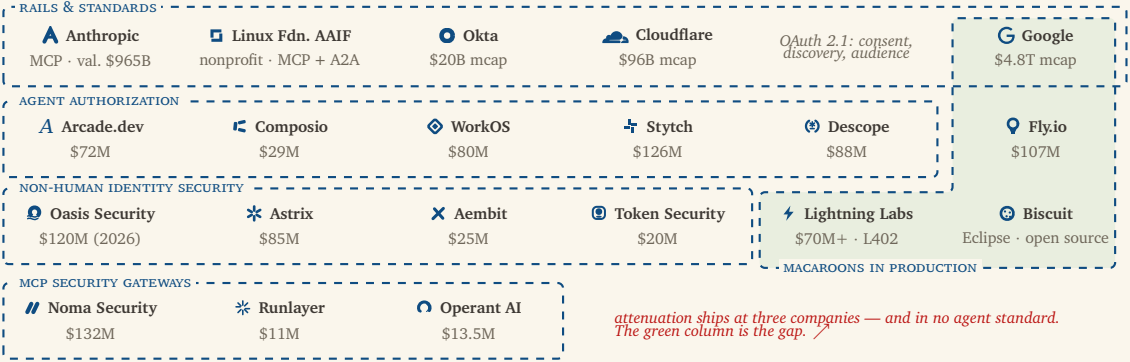
RFC 8693; MCP authorization spec (2025-11-25); Google DeepMind, arXiv 2602.11865 §4.7

The solution: macaroons

> Mint$(K, id) \to M_0$ — **verifier only; K never leaves**
$\text{sig}_0 = \text{HMAC}(K, id)$; $M_0 = (id, [\,], \text{sig}_0)$
> Grammar $\to \{P_j\}$ — **published once, static**
the predicates the verifier can evaluate: expires<T · tool∈S · slot∈S · session=s · …
> Attenuate$(M_i, c) \to M_{i+1}$ — **any holder, offline, no server contact**
$\text{sig}_{i+1} = \text{HMAC}(\text{sig}_i, c)$; append c to the list. Removing a caveat = an HMAC preimage ⇒ authority is **monotone decreasing** — a theorem, not a policy
> Verify$(M, \text{req}) \to \{0, 1\}$ — **root key, one pass**
replay $h := \text{HMAC}(K, id)$, then $h := \text{HMAC}(h, c_i)$ for all i; accept iff $h = \text{sig}$ *and* $\bigwedge_i P_{c_i}(\text{req}, \text{ctx})$ — an unknown caveat evaluates to **false** (fail closed)

*Authorization by possession, not identity: the verifier never asks who is calling — only that every caveat holds. Identity, if wanted, is just another caveat (*session=*s); expiry is the backstop.*

Ecosystem map I — locking credentials away from agents



The shape of a zero-knowledge problem

1 · no truth needed

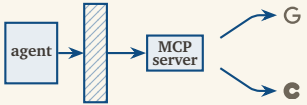
- agent acts on data as-is
- isolation & caveats suffice

2 · truth, shown whole

- signature moves the fact
- 3rd party or chain signs

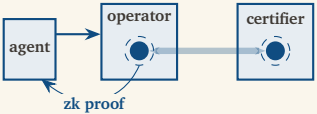
3 · truth, but hidden

- reveal predicate, not data
- issuer ≠ prover ≠ verifier



the calendar case — true or not, it just acts

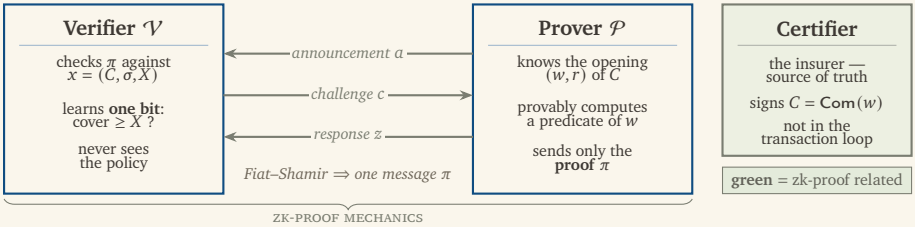
= Part 1 — the signature moves the fact, whole



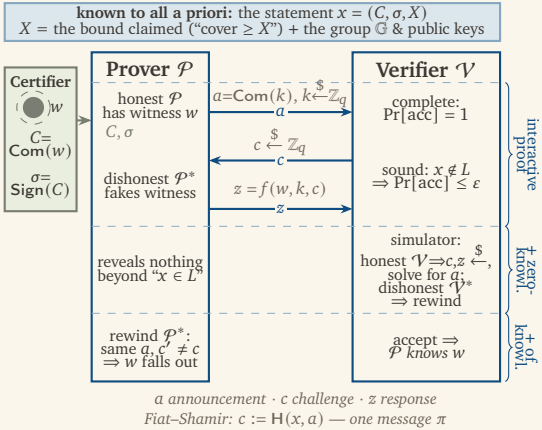
Three types of problems: No "truth" proof needed; revealing "truth" is ok; Truth proof is needed, and the truth cannot be revealed
— zk only sensible in the last case: issuer = prover ⇒ garbage.

Prove you're insured — without showing the policy

Scenario: the buying agent needs assurance of sufficient liability cover — the merchant will not reveal the whole policy.



Zero knowledge with an external certifier



The ingredients — recall

> **A policy** w
certified by the insurance company contains sensitive information (total coverage, premium)

> **Hide most of w**:
How do I extract what I want (coverage > X) and convince the verifier?

> **Make it usable**:
Convice the Verifier of correct calculation
⇒ **SNARK**

ZK needs an external commitment
— prover = issuer ⇒ garbage.

Goldwasser–Micali–Rackoff (1985/89); Thaler, §1.3; Fiat–Shamir (1986)

Selective disclosure of 3rd party certified facts

Proof step	Move	Who	What	Formula
#1 certify insurance policy	*commit*	certifier	Pedersen	$r \xleftarrow{\$} \mathbb{Z}_q;\quad C := g^w h^r$
	sign	certifier	any Part-1 scheme	$\sigma := \mathsf{Sign}_{sk}(C)$
	hand over	certifier $\to \mathcal{P}$	private channel, once	the credential (w, r, σ)
	check σ	$\mathcal{V}$	signature check pins C	$\mathsf{Verify}_{pk}(C, \sigma) \stackrel{!}{=} 1$
#2 $\mathcal{P}$ knows policy	*announce*	$\mathcal{P} \to \mathcal{V}$	commit to fresh nonces	$k_w, k_r \xleftarrow{\$} \mathbb{Z}_q;\quad a := g^{k_w} h^{k_r}$
	challenge	$\mathcal{V} \to \mathcal{P}$	uniform challenge	$c \xleftarrow{\$} \mathbb{Z}_q$
	respond	$\mathcal{P} \to \mathcal{V}$	fold the witness in	$z_w := k_w + c\,w;\quad z_r := k_r + c\,r$
	check	$\mathcal{V}$	one equation in $\mathbb{G}$	$g^{z_w} h^{z_r} \stackrel{?}{=} a \cdot C^c$

Pedersen (1991); Schnorr (1991); Okamoto (1992)

The exact math — proving cover $\geq X$ (a range proof)

PROOF STEP	MOVE	WHO	WHAT	FORMULA
#3 $\mathcal{P}$ proves the bound $w \geq X$	*shift*	both	public arithmetic	$D := C/g^X = \mathsf{Com}(d; r)$, $d = w - X$
	decompose	$\mathcal{P}$ privately	schoolbook binary	$d = \sum_{i<n} d_i 2^i$, $d_i \in \{0, 1\}$
	commit bits	$\mathcal{P} \to \mathcal{V}$	Pedersen, r_i tuned to r	$C_i := g^{d_i} h^{r_i}$, $\sum_i 2^i r_i = r$
	bits are bits	$\mathcal{P} \leftrightarrow \mathcal{V}$	OR proof per bit (↓)	each C_i opens to 0 or 1
	recombine	$\mathcal{V}$	one product in $\mathbb{G}$	$D \stackrel{?}{=} \prod_i C_i^{2^i}$ $\left(= g^{\sum d_i 2^i} h^r\right)$
	conclude	$\mathcal{V}$	size argument: $2^n \ll q$	$\sum d_i 2^i \leq 2^n - 1 \Rightarrow d \geq 0 \Rightarrow w \geq X$

The OR trick — one bit, two runs

claim: $d_i = 0$ **or** $d_i = 1$ — prove *both* at once:

real proof on the true side, *simulator* fakes the false

cost: a, c, z each become a pair $(\cdot_0, \cdot_1)$

lock: $c_0 + c_1 = c$ ($\mathcal{V}$'s challenge) — only *one* side can be pre-faked

Why the bits force $w \geq X$

- $D = \prod_i C_i^{2^i} \Rightarrow d \equiv \sum_i d_i 2^i \pmod q$
- n non-neg. bits $\Rightarrow 0 \leq d \leq 2^n - 1 \ll q$ (below the wrap)
- too small to have wrapped: $d = w - X \geq 0$, i.e. $w \geq X$

(a fake $w < X$ gives $d \approx q \gg 2^n - 1$)

Cramer–Damgård–Schoenmakers (1994); Bünz et al., Bulletproofs (2018)

The exact math — proving cover $\geq X$ (a range proof)

PROOF STEP	MOVE	WHO	WHAT	FORMULA
#3 $\mathcal{P}$ proves the bound $w \geq X$	*shift*	both	public arithmetic	$D := C/g^X = \mathsf{Com}(d; r)$, $d = w - X$
	decompose	$\mathcal{P}$ privately	schoolbook binary	$d = \sum_{i<n} d_i 2^i$, $d_i \in \{0, 1\}$
	commit bits	$\mathcal{P} \to \mathcal{V}$	Pedersen, r_i tuned to r	$C_i := g^{d_i} h^{r_i}$, $\sum_i 2^i r_i = r$
	bits are bits	$\mathcal{P} \leftrightarrow \mathcal{V}$	OR proof per bit (↓)	each C_i opens to 0 or 1
	recombine	$\mathcal{V}$	one product in $\mathbb{G}$	$D \stackrel{?}{=} \prod_i C_i^{2^i}$ $\left(= g^{\sum d_i 2^i} h^r\right)$
	conclude	$\mathcal{V}$	size argument: $2^n \ll q$	$\sum d_i 2^i \leq 2^n - 1 \Rightarrow d \geq 0 \Rightarrow w \geq X$

The OR trick — one bit, two runs

claim: $d_i = 0$ **or** $d_i = 1$ — prove *both* at once:

real proof on the true side, *simulator* fakes the false

cost: a, c, z each become a pair $(\cdot_0, \cdot_1)$

lock: $c_0 + c_1 = c$ ($\mathcal{V}$'s challenge) — only *one* side can be pre-faked

Why the bits force $w \geq X$

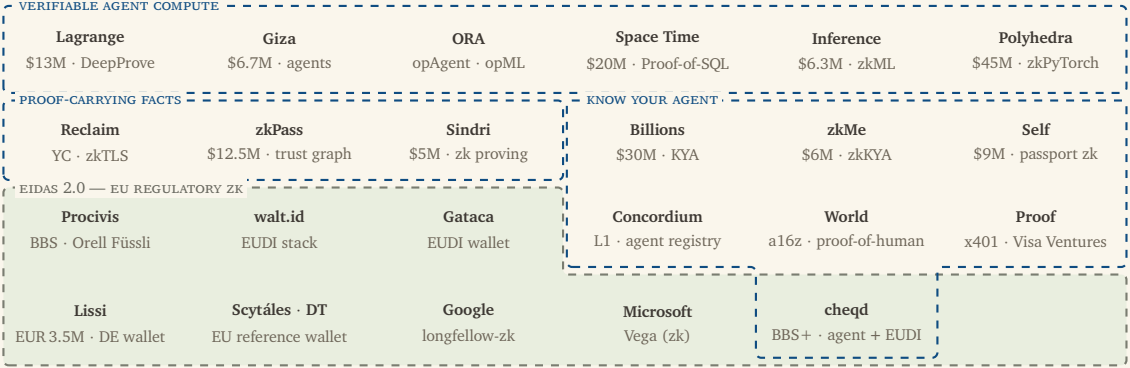
- $D = \prod_i C_i^{2^i} \Rightarrow d \equiv \sum_i d_i 2^i \pmod q$
- n non-neg. bits $\Rightarrow 0 \leq d \leq 2^n - 1 \ll q$ (below the wrap)
- too small to have wrapped: $d = w - X \geq 0$, i.e. $w \geq X$

(a fake $w < X$ gives $d \approx q \gg 2^n - 1$)

Homework: turn this into a SNARK — arithmetic circuit → R1CS constraints → QAP (zeros ↔ witness vector) → polynomial decomposition → check equality at random points

Cramer–Damgård–Schoenmakers (1994); Bünz et al., Bulletproofs (2018)

Ecosystem map II — zero knowledge for the agentic economy



 funding: company PRs & financial press, checked 2026-07-13; eIDAS 2.0 ZKP mandate; ARF Topic-G (2025)

Part 3

Compute as a Commodity

When an agent buys, it buys compute

Price — buy or build, in tokens

- agents buy what they **may not** or **cannot** do, are **bad at** — or can't make **more cheaply**
- scarcity is **context-dependent**: buy-or-build is traded off inside the agent's operating context
- the natural **unit of account is the token**: build cost ≈ token price × tokens to replace the service call
- paying **more** can be optimal — risky context, reliability required

Capacity — reserve now, consume later

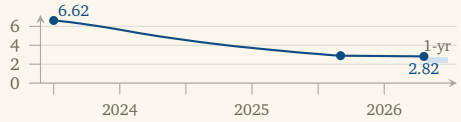
- an agent may also need to **reserve the right** to consume a resource later
- as in commodity markets: owners compensated for future **delivery**; both sides gain **predictability**
- compute isn't uniform (LLM, VM, GPU): a model on an infrastructure should **price by supply and demand**
- **spot** ≈ cloud provisioning today; **forward** = reserved future instances → compute **exchanges**

Compute as a commodity — unit, price, futures market

Unit I: the token — $/M tokens

usage-weighted market price per million LLM tokens

Unit II: the GPU-hour — $/hr



H100 rental composite; dots verified, line approximate

A futures market for compute

- **capacity in the future**: delivery later, price agreed now
- **cash-settled**: pays the move, up or down
- **physical**: access later, price set now

Settlement grades

- **compute is not compute**: settled by grade — WTI vs Brent, never "oil"
- token: **model**, version, context, latency
- GPU-hr: **chip**, memory, network, region

 SDLLMTK via Bloomberg Opinion; SemiAnalysis GPU index — curves approximate, checked 2026-07-13; CME & ICE, May 2026

The settlement questions

Cash — or physical?	Cash pays an index: a benchmark nobody can bend — the **oracle problem**. Physical is the point: agents must eventually **run** somewhere.
Was it there?	The **grade**: TEE **attestation** — the chip signs what it is. The reservation: **proof-of-capacity** heartbeats against the promised fleet.
What if it wasn't?	Seller posts collateral before selling: **staking** + **slashing** — burned or paid to the buyer on failure. A clearinghouse margin, decentralized.
How do I get in?	Delivery = a **scoped, attenuable credential** (macaroons, Part 2). Resale = handing over the token — authority only ever shrinks.
Who allocates? Who meters?	At delivery a **reverse auction** matches workloads to machines; **escrowed streaming payment** (x402, Part 1) meters capacity as it is consumed.
Same GPU-hour?	Grades settle by **benchmark-indexed differentials**, as oil settles quality. Paid on execution? **Verifiable compute**: refereed re-run, zkML.

Did I get what I paid for? — a market of certifiers

root of trust: silicon — the chip vendor signs	
genuine chip, real memory	quote by a **key fused in the chip**, chained to the vendor root
firmware patched	vendor **TCB status**: is this quote from an up-to-date chip?
what actually ran	**launch measurement** — hash of the booted image, inside the quote
isolation on, debug off	report **flags** — debug mode ⇒ walk away
root of trust: stake — a checker network signs, collateral at risk	
capacity really reserved	**liveness heartbeats**; lie ⇒ stake slashed (slide 3)
fabric, delivered speed	**benchmark oracles**, random spot re-runs
root of trust: law — auditors & regulators sign	
where, under which regime	certificates about the **building**, not the chip → next slide

Every fact needs its own issuer — the TEE's gift is a **certifier in the chip**; a signed fact is what ZK can hide (Part 2).

Who computes, and where?

Export controls

- **box**: EAR chips
- **session**: RASA — rent = export
- **model**: Fable 5

Sanctions

- **addresses** listed: Tornado Cash
- **mid-term listing**
- **gate at delivery**

Data localisation

- **GDPR**: access = transfer
- **hard**: CN, RU
- C5, SecNumCloud

AI rules

- **duty**: 10^{25} FLOP
- **extraterritorial** by output
- CN: **file** to serve

KYC meets settlement

- by **novation**: clearing house does KYC
- by **math**: ZK — proofs, not passports

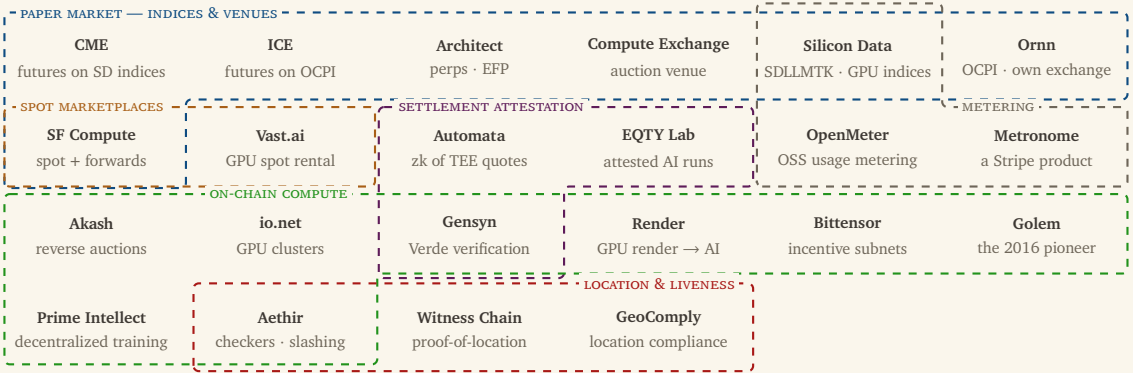
Proof of location

- by **silicon**: chips attest location (Chip Sec. Act)
- by **law**: auditors certify the building

finance-gated compute ? **permissionless delivery**

RASA H.R.2683; Fable 5/Mythos lifted 07/26; Chip Security Act H.R.3447; EU AI Act; GDPR; OFAC–Tornado Cash —

Ecosystem map III — the compute trading stack



company facts: PRs & docs, checked 2026-07-14 (see figure header); tiles in two clusters = bridges; no funding tags by design

Thank you

Trusting agents is a cryptography problem

Christian Pfrang

info@christianpfrang.com