MANDATE

A Verifiable-Mandate Execution Layer for Private, Policy-Constrained Agentic Trading

Protocol design document — v0.1 draft, July 2026

One-line thesis: Everyone is building either (a) private execution, (b) encrypted mempools, (c) agent wallets with plaintext policy engines, or (d) intent/solver networks. Nobody has built the thing that connects them: a layer where an autonomous agent can trade a *hidden* strategy over a *hidden* portfolio while producing *public, succinct, composable proofs* that every action complied with a pre-committed risk mandate — and where those proofs accumulate into a portable, privacy-preserving reputation that lets strangers delegate capital to agents they cannot inspect. MANDATE is that layer. It is deliberately **not** a new L1. It is a cryptographic middleware + app-specific settlement layer anchored to Ethereum.

A. Research Map

The comparison below is organized by what each system is *actually for*, because "L1 vs L2" obscures the relevant tradeoffs. Status notes reflect mid-2026.

A.1 General-purpose base layers

Chain	Consensus	Execution	DA	Privacy	Fee model	Key strength	Key weakness	Why it hasn't solved agentic trading
Bitcoin	Nakamoto PoW	UTXO script (non-Turing-complete)	On-chain	Pseudonymous only	Fee auction	Maximal credible neutrality, deepest security budget	No expressive execution; 10-min blocks	Cannot express policies, auctions, or proofs natively
Ethereum	Gasper PoS (LMD-GHOST + Casper FFG)	EVM, single-threaded, global state	On-chain + blobs (EIP-4844)	None at L1 (encrypted mempool EIPs in draft; EIP-8105 discussion ongoing)	EIP-1559 base+tip	Deepest liquidity, best tooling, credible settlement	12s slots, full transparency, MEV supply chain centralized around a	Transparency leaks strategies; latency too high; MEV taxes

Chain	Consensus	Execution	DA	Privacy	Fee model	Key strength	Key weakness	Why it hasn't solved agentic trading
							few builders	every agent
Solana	Tower BFT + PoH	Sealevel parallel runtime (SVM)	On-chain (validators need high bandwidth)	None; mempool-less (Gulf Stream) but leaders/Jito see flow	Local fee markets + priority fees	~400ms slots, cheap, high throughput	Validator hardware centralization; MEV moved into Jito bundle auctions rather than disappearing; occasional liveness incidents historically	Speed without privacy = fastest copy-trading machine ever built; leaders see everything
Avalanche	Snow* metastable consensus	EVM (C-chain) + custom VMs; L1s (former subnets)	Per-chain	None	Per-subnet	Sub-second finality, app-chain flexibility	Fragmented liquidity across subnets; validator economics for small subnets weak	App-chains isolate liquidity, the one thing traders need pooled
Cosmos	CometBFT	CosmWasm / custom SDK modules	Per-chain	None (except Secret)	Per-chain	Sovereign app-chains, IBC is the most battle-tested trust-minimized messaging	Every chain bootstraps its own validator security; liquidity fragmentation	Same as Avalanche; dYdX v4 shows an orderbook app-chain works, but is fully transparent and validator-front-runnable in principle

Chain	Consensus	Execution	DA	Privacy	Fee model	Key strength	Key weakness	Why it hasn't solved agentic trading
Polkadot	BABE/GRANDPA, shared security	Parachains (WASM)	Relay chain	None native	Coretime	Shared security model was ahead of its time	Developer mindshare declined; agile coretime underused	Ecosystem gravity insufficient for a liquidity-dependent use case
Near	Nightshade sharded PoS	WASM, async runtime	Sharded	None	Predictable gas	Sharding actually shipped; chain-abstraction/intents push (Near Intents)	Async composability is hard for atomic trading logic	Cross-shard async breaks atomic multi-leg trades
Aptos / Sui	AptosBFT / Narwhal-Bullshark + Mysticeti	Move VM (parallel, object-centric on Sui)	On-chain	None	Low, storage-priced	Sub-second finality, parallel execution, Move's resource safety	Younger security track record; liquidity depth far below ETH/SOL	Fast and transparent — same structural problem as Solana

A.2 Modular / infrastructure layers

Project	What it is	Security assumption	Strength	Weakness	Relevance to MANDATE
Celestia	DA layer with data availability sampling (DAS)	Honest-minority light clients can verify DA via sampling; 2/3 honest for consensus	Cheap, scalable blob space; DAS is genuinely elegant (O(√n) sampling per light node against erasure-coded data)	DA-only; no execution, no settlement; token value accrual unclear	Candidate DA layer for encrypted order-flow blobs
EigenLayer / restaking	Rent Ethereum stake as economic security for AVSs	Slashing conditions must be objectively attributable; cryptoeconomic, not cryptographic, security	Bootstraps security for middleware (oracles, DA, keyper committees) without a new token	Slashing-condition design is hard; correlated-slashing risk; "security as a service" is only	Natural home for MANDATE's keyper committee and watchtower network

Project	What it is	Security assumption	Strength	Weakness	Relevance to MANDATE
				as good as the fault attribution	
Espresso / shared sequencers	Shared sequencing + fast confirmations for rollups	BFT among sequencer set	Cross-rollup atomicity potential	Adoption chicken-and-egg; MEV just moves to the shared sequencer	Optional future integration for cross-rollup batch auctions

A.3 Privacy systems

Project	Privacy tech	Model	Strength	Weakness	Why it doesn't solve this problem
Zcash	Groth16 → Halo 2 (no trusted setup) shielded pool	Private payments (UTXO notes)	Oldest production zk system; Halo 2 recursion pioneered here	Payments only; low shielded adoption; no programmability	Can't express strategies, mandates, or markets
Monero	Ring signatures + RingCT + stealth addresses	Probabilistic anonymity sets	Default-on privacy, real usage	Anonymity set is statistical, not cryptographic; repeated deanonymization research; no smart contracts	Same
Aztec	Client-side zk proofs (UltraHonk), Noir language, hybrid private/public state	Programmable privacy L2 on Ethereum; Ignition mainnet live since Nov 2025, transaction execution enabled during 2026; 36–72s blocks at launch, targeting ~4s	Best-in-class programmable privacy; decentralized sequencers/provers from day one	Latency (seconds at best), young ecosystem, private-state UX (note discovery, encrypted state sync) still rough; general-purpose, so no trading-specific market structure	Closest neighbor. But Aztec gives you *general* private execution — it does not give you batch-auction market structure, policy-compliance proofs against externally anchored portfolio state, or portable agent reputation. You could build parts of MANDATE *on* Aztec (see §L)
Aleo	zkVM (snarkVM), off-chain execution + on-chain verification, Leo language	Privacy L1	Full-program privacy; own consensus (AleoBFT)	Own L1 = liquidity island; proving costs for complex programs; ecosystem small	Liquidity isolation is fatal for trading

Project	Privacy tech	Model	Strength	Weakness	Why it doesn't solve this problem
Secret Network	TEE-based (SGX) encrypted contracts	Cosmos chain, hardware trust	Practical, fast private compute since 2020	SGX has a long CVE history (ÆPIC, SGAxe, Downfall...); hardware root of trust is a single vendor	Demonstrates TEE tradeoff: fast but trust-heavy — MANDATE uses TEEs only as an *optional* latency tier, never as the root of trust
Starknet	STARK validity proofs (Cairo VM)	Scaling-first L2; privacy via client-side proving on roadmap	Transparent setup, post-quantum-plausible hashes, fast recursive proving (Stwo)	Privacy not the default; auditor viewing-key model where it exists	Proof tech is relevant; the network is not privacy-native
Penumbra	Threshold-encrypted DEX (batch swaps) on Cosmos	Shielded chain-level DEX	*The* prior art for private batch-auction trading: sealed flow, batch clearing prices	Own chain (liquidity island), payments/swap-focused, no agent policy layer, no portable reputation	Proves the market-structure half of the thesis works; missing the agent/mandate/reputation half and Ethereum liquidity

A.4 MEV, intents, and agent infrastructure (the direct competitive frontier)

System	What it does	Status (mid-2026)	What's missing for agentic trading
Flashbots / SUAVE line of work	PBS, MEV-Share, TEE-based builders (BuilderNet), programmable privacy for order flow	TEE block-building in production; SUAVE's grand unified vision partially absorbed into BuilderNet-style infra	Infrastructure for *block builders*, not a policy/verification layer for *capital delegators*
Shutter / Primev encrypted mempool	Threshold-encrypted transactions, keypers release keys post-inclusion; live on Gnosis, FastRPC lane on Ethereum PBS (PoC late 2025 → mainnet), EIP-8105 "Universal Enshrined Encrypted Mempool" drafted for a future hard fork	Production and advancing	Encrypts *transactions in transit*. Does nothing about strategy privacy at rest, portfolio privacy, policy compliance, or reputation. MANDATE consumes this as a component
CoW Protocol	Batch auctions with uniform clearing prices, solver competition	Mature; the strongest deployed	Order *intent* is public once in the auction; no private portfolio, no agent mandates

System	What it does	Status (mid-2026)	What's missing for agentic trading
		MEV protection for swaps	
ERC-7683 / Open Intents Framework	Standard cross-chain intent struct + settlement interface; adopted by 30+ projects; solvers share inventory across protocols	The de facto cross-chain execution rail	A message format, not a trust layer. Says nothing about who the agent is, whether it's allowed to sign this intent, or whether it's behaving
Anoma	Intent-centric L1, counterparty discovery + multi-intent matching, mainnet targeted late 2026	Pre-mainnet	Beautiful generality; but a new L1 with new liquidity, and intents in its public gossip layer face their own information-leakage problems
Agentic wallets (Cobo Pacts, session-key AA stacks, ERC-7521 ecosystems)	Policy engines at the wallet layer: spending caps, whitelists, session keys, human-in-loop approvals; TEE/MPC custody	Commercially real in 2026	**Policies are enforced in plaintext by an operator you must trust, verified by nobody, and portable to nowhere.** A Cobo Pact protects *you* from *your* agent. It does not let a *stranger* verify an agent's history and safely delegate to it. This is exactly the trust gap MANDATE's proofs close
ZKML (Modulus-style, Giza, verifiable inference)	Prove a specific model produced a specific output	Niche production use	Proves the *wrong thing* at the wrong cost: you rarely care that inference was faithful (proving a transformer forward pass costs orders of magnitude more than the trade); you care that the resulting *action* respected constraints. MANDATE proves the action, not the model
Oasis, Phala, TEE agent platforms	Run agents inside attested TEEs; verifiable-agent trading bots (e.g., Oasis's 2025 launch)	Live	Attestation says "this binary ran on this CPU." It inherits Intel/AMD trust, side-channel risk, and reveals nothing composable about *financial* behavior over time. Useful as MANDATE's optional fast lane, insufficient as its foundation

A.5 Cryptographic primitives — tradeoffs

For each primitive: what it gives you, what it costs, and its trust base. Sizes/timings are order-of-magnitude for 2026-era implementations.

zk-SNARKs (Groth16, Plonk/UltraHonk families). Succinct arguments of knowledge for NP relations. Groth16: proofs ~128–192 bytes (2–3 group elements on BN254), verification 2–3 pairings (~200–300k gas on Ethereum before batching), but a *per-circuit* trusted setup. Plonkish/Honk: universal setup (one structured reference string, e.g. perpetual powers-of-tau), proofs ~400B–2KB, verification a few hundred k gas, flexible custom gates and lookups. Security: knowledge-of-exponent / algebraic-group-model

assumptions plus discrete log on pairing-friendly curves (BN254 ≈ 100–110 bits after TNFS improvements; BLS12-381 ≈ 120+). Prover cost is the real constraint: dominated by multi-scalar multiplications, roughly O(n log n) in constraint count n. A small policy circuit (10^3–10^5 constraints) proves in **tens of milliseconds to ~1 second on a laptop** — this single fact makes MANDATE feasible.

zk-STARKs. Hash-based (FRI) IOPs: transparent setup, plausibly post-quantum, blazing provers (especially with small fields — Baby Bear / Mersenne-31 in Plonky3/Stwo lineage), but proofs are 50–200KB, so on-chain verification needs either a recursive SNARK wrap or expensive calldata. Right pattern: **STARK-prove fast off-chain, wrap in a SNARK for on-chain verification.** MANDATE uses exactly this for its recursion layer.

FHE. Compute on ciphertexts without decrypting (TFHE, CKKS, BGV). The honest number: even with 2025–26 hardware acceleration, TFHE bootstrapping is ~1–10ms *per gate*; an order-matching engine over encrypted books is 10^4–10^6× slower than plaintext, and threshold-FHE decryption committees reintroduce collusion assumptions anyway. FHE also gives you *no integrity* by itself — you need zk proofs on top to know the encrypted computation was done correctly. Verdict: architecturally beautiful, not deployable for latency-sensitive trading in 2026. MANDATE excludes it from the core and names it a research-track upgrade for encrypted matching (§J).

MPC. Secret-shared computation among n parties, secure if fewer than t collude (honest-majority protocols are fast-ish; dishonest-majority with malicious security costs heavy preprocessing). Communication rounds dominate latency — every multiplication gate needs interaction, so WAN latency × circuit depth kills sub-second trading. Where MPC *is* the right tool: threshold key management (t-of-n signing for agent accounts), distributed key generation for the encryption committee. MANDATE uses MPC there and only there.

TEEs (SGX/TDX/SEV-SNP, Nitro). Native-speed private compute with remote attestation. Trust base: the CPU vendor's key hierarchy + microcode + absence of side channels — historically violated repeatedly (Foreshadow, SGAxe, ÆPIC Leak, Downfall). Correct usage pattern in 2026: **defense-in-depth accelerator, never sole root of trust.** MANDATE's design principle: TEE compromise may leak *confidentiality* (strategy privacy degrades) but must never break *integrity* (funds and mandate enforcement stay safe, because those rest on zk proofs and on-chain state).

Threshold signatures (BLS, FROST/Schnorr, GG-style ECDSA). t-of-n signing where no party holds the key. BLS aggregates beautifully (one pairing check for many signers). Used in MANDATE for: keyper committee decryption shares, watchtower attestations, and optionally agent-account custody.

Threshold / identity-based encryption for mempools. Encrypt to a batch-ID "identity"; a keyper committee releases the decryption key only after ordering is fixed (Shutter model, pairing-based IBE). Assumption: fewer than t keypers collude; mitigations emerging include traitor-tracing schemes and silent-setup threshold encryption (Garg et al.) that removes the DKG bottleneck and makes committees dynamic. This is MANDATE's order-flow privacy layer, consumed rather than reinvented.

VDFs. Sequential-computation time locks (Wesolowski/Pietrzak over class groups or RSA). Community consensus has shifted *away* from VDF-based delay encryption for mempools — hardware-speed uncertainty makes release timing sloppy versus committee-released keys. MANDATE follows that shift: no VDFs in the core; possible tie-breaker randomness source only.

Recursive proofs & folding (Halo2 accumulation, Nova/SuperNova/HyperNova, Plonky/Honk recursion). Verify a proof inside a circuit, or better, *fold* many instances into one running instance with

only a final SNARK at the end (Nova folding: each step costs ~one MSM rather than full verification). This is the enabling technology for MANDATE's **reputation kernel**: thousands of per-epoch compliance proofs compress into one constant-size credential. IVC (incrementally verifiable computation) is precisely the "proof-carrying history" abstraction reputation needs.

Data availability sampling. Erasure-code data 2×, light clients sample O(√n) or O(polylog) chunks; with enough samplers, withheld data is detected w.h.p. Relevant for publishing encrypted order-flow blobs cheaply (Celestia or Ethereum blobs) so settlement is reconstructible without a data monopoly.

Account abstraction (ERC-4337 / EIP-7702). Programmable validation: the account itself decides what a valid "signature" is. This is the on-chain enforcement point for mandates — the account contract *refuses* any agent action lacking a valid compliance proof or exceeding session-key scope. EIP-7702 lets EOAs adopt this behavior. Session keys give agents narrow, expiring, revocable authority.

Intent architectures & shared sequencing. Covered in A.4; MANDATE treats ERC-7683 as its cross-chain execution rail and batch auctions as its ordering discipline.

Privacy-preserving identity/reputation (Semaphore-style nullifiers, BBS+ credentials, zk attestations). Prove group membership or credential possession without linkability; nullifiers prevent double-use. MANDATE binds reputation to a persistent agent identity commitment while keeping trade history unlinkable — the classic anonymity-vs-accountability tension resolved by *pseudonymous accountability*: one stable pseudonym, provable history, hidden actions.

B. Gap Analysis

What remains unsolved at the intersection, stated as sharply as possible:

1. The delegation-trust gap (the core gap). In mid-2026 you can (a) run an agent in a TEE, (b) cage it with a plaintext wallet policy, (c) route its orders through an encrypted mempool, and (d) settle cross-chain via ERC-7683. What you *cannot* do is let a stranger with $50k verify — cryptographically, without trusting you, your cloud, or Intel — that your agent has operated for six months, never breached a 2% daily-loss limit, never touched an unwhitelisted asset, and achieved returns in a stated band, *without revealing a single trade*. Wallet policies aren't verifiable by third parties; TEE attestations aren't financial statements; on-chain history is verifiable but fully doxxing. Delegation to autonomous agents therefore still runs on brand trust — exactly the thing crypto exists to remove.

2. Strategy privacy vs. accountability. Transparent chains make every profitable agent a free API for copy-traders and an oracle for adversarial MEV; private chains (Aleo, shielded pools) make agents unauditable black boxes. No deployed system provides the middle point: *hidden actions, provable constraints*.

3. Portfolio-state-conditioned policies. Real risk limits are stateful — drawdown from high-watermark, aggregate leverage, position concentration. Enforcing them requires the enforcement layer to know the portfolio; publishing the portfolio destroys the agent. Today's answer is "trust the operator's risk engine." There is no system where the *state the policy is checked against* is itself cryptographically anchored (so the agent can't lie about it) yet private (so observers learn nothing).

4. Reputation portability. Agent reputation today = a token price (Virtuals-style), a Twitter following, or a platform-local score. None survives moving to a new venue, none is privacy-preserving, and most are trivially gameable. Nothing compresses verified behavioral history into a portable credential.

5. MEV protection at agent scale. Encrypted mempools (in production) protect the *transit* of a transaction. Frequent batch auctions (CoW, Penumbra) protect *ordering*. But agents leak alpha through a wider channel set: position changes visible post-settlement, timing patterns, size fingerprints, venue selection. No system addresses the post-trade leakage surface for a *repeated player*, which is what an agent is.

6. Latency vs. proofs. Validity proofs take milliseconds-to-seconds; HFT takes microseconds. This gap is physics, not engineering, at current proving costs. The unsolved (actually: unsolvable-head-on) problem becomes a *design constraint*: build market structure where sub-second speed doesn't win — batch auctions — instead of pretending to out-race Solana.

7. Interoperability of trust, not just messages. ERC-7683 standardized intent *messages*. Nobody standardized intent *authority*: a proof-carrying answer to "is this agent allowed to sign this, and is it in good standing?" that any solver or settlement contract on any chain can check in one verification.

Gaps 1, 3, 4, and 7 are the whitespace. Gaps 2, 5, 6 shape the design.

C. Design Principles

P1 — Prove actions, not algorithms. Never prove "the model is X" (ZKML's dead end). Prove "the action satisfied constraint set P over anchored state S." Constraint circuits are small; model circuits are enormous. This single choice makes the whole system cheap.

P2 — Integrity from cryptography, confidentiality from layered defense. Funds safety and mandate enforcement must rest only on zk proofs + on-chain state (cryptographic). Strategy secrecy may additionally lean on threshold encryption and optional TEEs (economic/hardware). A total TEE break must cost privacy, never money.

P3 — The state you prove against must not be self-reported. The portfolio state root is maintained by the settlement contract as a function of settled fills. The agent proves compliance against *that* root. An agent cannot fabricate a healthy portfolio, because it doesn't control the commitment.

P4 — Don't race; batch. Adopt frequent batch auctions with uniform clearing prices as the native market structure. Within a batch, arrival order is irrelevant → latency competition and front-running lose their payoff function, and proof-generation time (hundreds of ms) fits inside the batch interval by construction.

P5 — Middleware, not a new chain. Liquidity, security, and users live on Ethereum and its rollups. A new L1 restarts all three from zero and dies of loneliness. MANDATE is contracts + circuits + a thin operator network, restaking-secured, settling to Ethereum, executing cross-chain via ERC-7683.

P6 — Consume commodity components. Encrypted mempool tech (Shutter-style threshold IBE), intent standards (7683), AA (4337/7702), DA (blobs/Celestia), proof systems (Noir/Honk, folding) all exist and

are hardening. MANDATE's originality budget is spent entirely on the mandate/reputation layer — the part nobody has built.

P7 — Pseudonymous accountability. One persistent agent identity commitment; unlinkable actions; provable aggregate history; principal-side instant revocation; opt-in selective disclosure (e.g., to a regulator or an allocator under NDA) via viewing keys held by the *principal*, not the network.

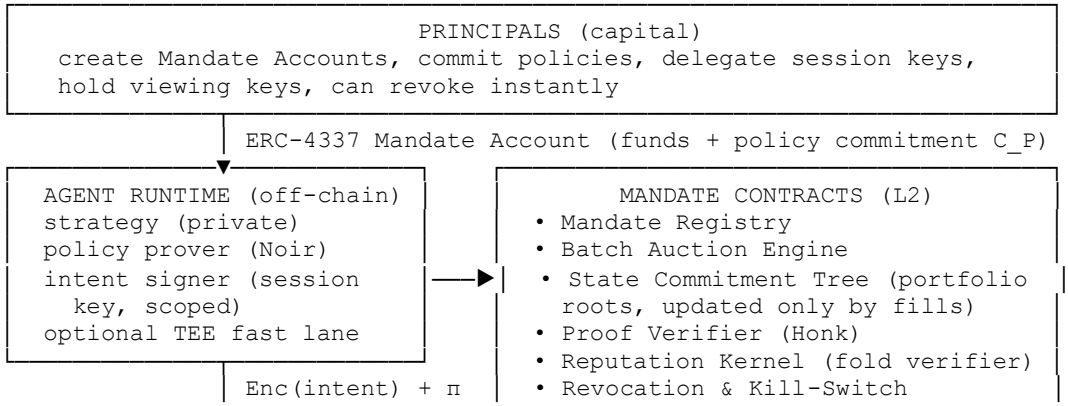
P8 — Graceful degradation and exits. Keyper failure → batches fall back to public-but-batched mode (ordering protection survives, transit privacy pauses). Prover failure → agent can halt but funds always exit via L1 escape hatch. Every trust assumption gets an explicit failure mode and a user-visible downgrade, never a silent one.

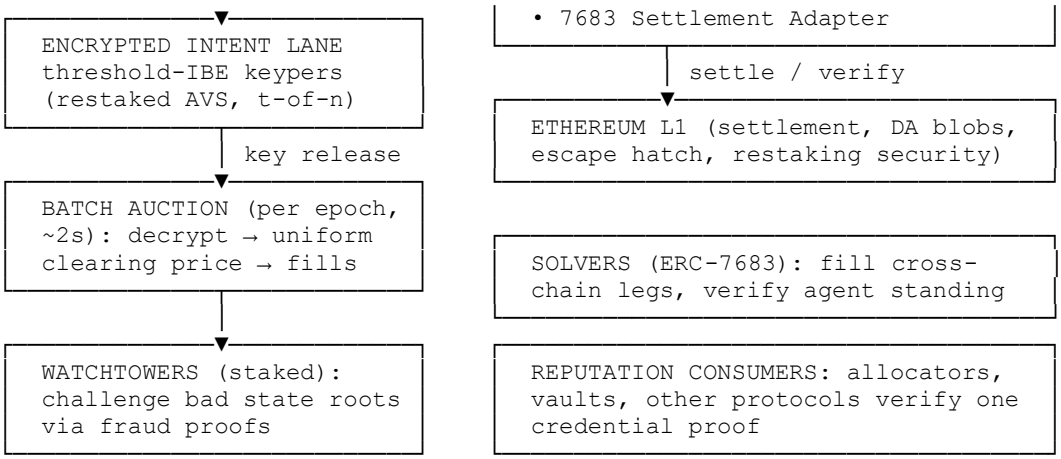
D. Final Project Idea

MANDATE is a cryptographic middleware and settlement layer on Ethereum where autonomous trading agents operate under **Verifiable Mandates**: risk policies committed on-chain as succinct commitments, enforced by the agent's smart account, and attested per-batch by small zero-knowledge proofs showing that each (still-encrypted, never-revealed) order complied with the mandate given the agent's on-chain-anchored but private portfolio state. Orders flow threshold-encrypted into frequent batch auctions with uniform clearing prices — neutralizing front-running, sandwiching, and copy-trading — and settle locally or cross-chain through ERC-7683 solvers who can verify an agent's authority and standing in a single proof check. Every batch's compliance proof is folded (Nova-style IVC) into a constant-size, ever-growing **Reputation Credential**: a portable, privacy-preserving, cryptographically unforgeable track record ("14,000 batches, 0 violations, drawdown never exceeded 4%, returns in bucket B3") that lets capital delegate to agents whose strategies it will never see. In one sentence: **MANDATE makes "trust me, my bot is safe" into a theorem.**

E. Technical Architecture

E.1 System components





E.2 Network layers

1. **Settlement & security:** Ethereum L1. Mandate contracts deploy on a low-cost L2 (or as an app-rollup later); final state roots and the escape hatch live at L1.
2. **Operator layer (thin):** keyper committee + watchtowers, both as restaked AVS operator sets with slashing (premature decryption evidence via traitor-tracing → slash; false/absent fraud challenges → slash/forfeit bond).
3. **Execution layer:** agents run anywhere. The protocol never sees the strategy — only encrypted intents and proofs.
4. **Liquidity layer:** native batch-auction pools per market, plus solver-bridged access to external liquidity (AMMs, orderbooks, other chains) via 7683.
5. **DA layer:** encrypted intent blobs + auction transcripts posted to Ethereum blobs (Celestia as a cost fallback), so any party can reconstruct and audit settlement.

E.3 Consensus / settlement model

No new consensus. Batch sequencing is deterministic given the encrypted intent set and the on-chain epoch boundary: intents included in epoch e are exactly those whose ciphertexts landed in the epoch-e blob before the cutoff (enforced by L2 inclusion, inheriting its ordering). Because ciphertexts are indistinguishable, *inclusion* ordering carries no exploitable information — the auction's uniform price makes intra-batch order irrelevant anyway. Settlement finality = L2 finality → L1 finality.

E.4 Execution environment

The **auction engine** is a smart contract (MVP) evolving into a proved-off-chain computation (research version): collect decrypted orders for epoch e, compute the uniform clearing price p* per market (maximize executed volume; standard call-auction rule: p* = price crossing max cumulative bid/ask overlap, ties broken by pro-rata), execute fills, update the state commitment tree. Agents' *strategies* execute entirely off-chain; the chain only ever sees ciphertexts, proofs, and fills.

E.5 Privacy layer (three concentric rings)

- **Ring 1 — transit privacy:** intents encrypted under threshold IBE to identity `epoch_id`; keypers release the decryption key only after the epoch's intent set is fixed on-chain. Pre-execution, nobody (including the sequencer) reads flow.
- **Ring 2 — state privacy:** portfolio state is a commitment tree: leaves are Poseidon commitments `com(position_i; r)` per (agent, asset). The contract updates commitments homomorphically-in-effect by verifying a small zk proof with each settled fill ("new leaf = old leaf + fill delta"), so the tree is *anchored on-chain but opaque*. Balances never appear in plaintext on-chain; the principal's viewing key decrypts locally.
- **Ring 3 — behavioral privacy:** uniform-price batches mean an individual fill can't be attributed to an individual agent from the public transcript (fills are settled against the pool at p*; per-agent deltas live only in commitment updates). Copy-trading requires breaking Ring 1 (collusion), Ring 2 (Poseidon), or side channels (§F.7).

E.6 Data availability

All ciphertexts + auction transcripts + commitment-tree diffs go to blob space. DAS-secured DA guarantees any watchtower can reconstruct state and challenge. Withheld data → epoch is challengeable → auction reverts to last good root (funds never at risk; liveness pauses).

E.7 Cross-chain communication

Outbound: an agent's intent may name a destination chain; after local auction clearing, the residual leg becomes an ERC-7683 `CrossChainOrder` filled by solvers, escrow-settled with the standard optimistic/oracle path of the chosen rail. Inbound: solvers and external protocols verify agent standing by checking one Honk proof against the Reputation Kernel root (relayed via a light-client or messaging rail — Hyperlane/IBC-style — as a read-only root, minimizing bridge trust to root relay).

E.8 Agent identity & reputation

- Identity = `id = Poseidon(pk_agent, salt)` registered once in the Mandate Registry; session keys authorized under it.
- Each epoch's compliance proof π_e is folded into a running instance (Nova-style folding over a cycle of curves; final wrap to Honk for cheap on-chain verification).
- The credential proves, over the *entire* folded history: epoch count, violation count (must be 0 by construction — violating intents are simply unprovable and thus never enter auctions), max drawdown bucket, notional volume bucket, and mandate-class hash. Buckets (range proofs) rather than exact values prevent statistical strategy reconstruction.
- Nullifier per (id, epoch) prevents proof reuse/double-counting.

E.9 Wallet / account abstraction

Mandate Account = ERC-4337 smart account with a validation rule: a user-operation from a session key is valid iff it carries (a) an in-scope action type, (b) an unexpired, unrevoked session, and (c) for trading actions, a verifying compliance proof referencing the current policy commitment C_P and current state root. Principals retain a superior owner key: instant revoke (takes effect next epoch — bounded damage = one epoch inside mandate limits, a quantifiable maximum loss), policy rotation (new C_P with timelock), withdraw-all to L1.

E.10 Risk control system

Defense in depth, in enforcement order:

1. **Unprovability (primary):** a non-compliant order cannot generate a valid proof → never enters the auction. Enforcement is *ex ante*, not punitive.
2. **Account-level caps:** session scope, per-epoch notional ceiling, venue mask — enforced in the account contract even if the proof layer had a bug.
3. **Watchtowers:** challenge incorrect state-root transitions (fraud-proof window, e.g. 10 epochs) — protects against auction-operator faults, not agent faults.
4. **Kill switch:** principal revocation; plus an automatic circuit-breaker leaf in every policy: `if oracle_drawdown_flag then only risk-reducing orders provable` (the policy circuit takes a signed oracle bit as public input).

E.11 MEV protection summary

Front-running/sandwiching: dead by Ring 1 + uniform price (nothing to see; nothing gained by ordering). Copy-trading: dead by Rings 1–3 for per-trade copying; degraded to slow statistical inference at best. Settlement-boundary MEV (cross-chain legs): inherited from the 7683 rail — mitigated by Dutch-auction fill pricing and solver competition, honestly not eliminated (§K). Keyper collusion: t-of-n + slashing + traitor-tracing; residual trust stated openly.

E.12 Fee model

- **Auction fee:** small bps on cleared notional, split: keypers (transit privacy) / watchtowers (state security) / protocol treasury.
- **Proof-verification gas:** amortized — the epoch's compliance proofs are aggregated (batch pairing check / recursion) so per-agent marginal verification cost is O(1) group operations.
- **Reputation queries:** free to verify (that's the point); optional paid attestation service for off-protocol consumers.
- **No protocol token required at MVP.** Fees in the settlement asset. A token, if ever, is for operator staking only — resisting the reflexive urge to tokenize is itself a design decision.

F. Cryptographic Architecture

F.1 Primitive selection and rationale

Function	Primitive	Why this one
Compliance & state-transition proofs	Plonkish SNARK (UltraHonk) over BN254, circuits in **Noir**	Universal setup; custom gates + lookups make range checks and Poseidon cheap; BN254 pairing precompiles make Ethereum verification ~250–350k gas, aggregatable; Noir is the most ergonomic circuit language with an active toolchain (Barretenberg)

Function	Primitive	Why this one
Commitments (state tree, policy, identity)	**Poseidon2** hash over BN254 scalar field	SNARK-native: ~1–2 orders of magnitude fewer constraints than SHA/Keccak in-circuit
Reputation accumulation	**Folding-scheme IVC** (Nova/HyperNova lineage over a 2-cycle of curves), terminal wrap to Honk	Folding cost per step ≈ one MSM, independent of history length; constant-size final proof; exactly matches "append-only verified history"
Order-flow encryption	**Threshold IBE** (Boneh–Franklin style over BLS12-381), t-of-n keyper committee, batch-ID as identity	Production-proven pattern (Shutter); encrypt-to-future-batch needs no per-recipient key exchange; silent-setup threshold schemes as upgrade path for dynamic committees
Committee & watchtower attestations	**BLS threshold signatures**	Aggregation → one pairing check regardless of committee size
Agent authority	ERC-4337 validation + **session keys** (ECDSA/secp256k1 for wallet compat; BLS optional)	Native to the Ethereum account stack
Selective disclosure	Principal-held **viewing keys** (ECIES to principal pk) + BBS+-style attestations for allocator-facing claims	Disclosure is a *user* right, not a network backdoor
Optional fast lane	**TDX/SEV-SNP TEEs** with remote attestation	Confidentiality accelerator only (P2); attestation quotes verified on-chain via an attestation-verification contract

F.2 Mathematical assumptions (explicit)

1. Discrete log & q-type assumptions (q-DLOG / AGM analyses) on BN254 and BLS12-381 pairing groups. Known concrete-security caveat: BN254 ≈ 100–110-bit after Kim–Barbulescu TNFS; acceptable for value-bounded, rotatable trading credentials, with a stated migration path to BLS12-381/Goldilocks-STARK verification as gas costs allow.
2. Bilinear Diffie–Hellman (for IBE).
3. Poseidon2 collision/preimage resistance in the algebraic setting (less cryptanalytic history than SHA-2 — named as residual risk).
4. Universal trusted setup: 1-of-N honesty in the powers-of-tau ceremony (perpetual ceremony, thousands of contributors).
5. Threshold honesty: <t of n keypers collude (economic + cryptographic backstops).
6. Standard rollup assumptions of the host L2 (or Ethereum directly).
7. **Not** assumed for fund safety: TEE integrity, keyper honesty (their failure leaks privacy, not funds), watchtower liveness (its failure delays, not loses).

F.3 The policy circuit (heart of the system)

Public inputs: policy commitment `C_P`, current state root `R_e`, epoch `e`, agent id commitment, intent commitment `com(o)` (matching the ciphertext's committed payload), oracle circuit-breaker bit + oracle

signature. Private inputs (witness): policy parameters P with opening of `C_P`; order o = (market, side, size, limit); portfolio leaves + Merkle paths to `R_e`; running high-watermark & daily PnL leaves.

Constraints (illustrative, ~30–80k constraints total):

- `C_P = Poseidon(P)`; Merkle memberships of touched leaves in `R_e`.
- Asset whitelist: `market ∈ MerkleSet(P.whitelist_root).`
- Notional: `size × limit ≤ P.max_order_notional`; `|position + Δ| ≤ P.max_position.`
- Leverage: `Σ|position_i|·price_i ≤ P.max_leverage × equity` (prices as signed-oracle public inputs).
- Drawdown: `HWM − equity ≤ P.max_drawdown × HWM` (range proofs via lookups).
- Rate limit: epoch order-count leaf `< P.max_orders_per_epoch`.
- Circuit-breaker: `breaker = 1 ⇒ sign(Δ·position) < 0` (risk-reducing only).
- Binding: `com(o)` opens to the same o encrypted in the ciphertext (proof of plaintext knowledge — ciphertext hash bound as public input; encryption performed with committed randomness).

Prover time: well under a second on commodity hardware; fits a 2s epoch with margin.

F.4 State-transition proof

Per settled epoch, the auction operator (contract in MVP; proved off-chain executor later) shows: `R_{e+1}` is `R_e` updated by exactly the cleared fills at p*, each fill consistent with a decrypted intent whose ciphertext was in the epoch blob and whose compliance proof verified. In the research version this is one recursive proof; in the MVP it's plain contract execution (transparent but trust-minimized), with commitments-only leaves preserving Ring 2.

F.5 Reputation folding

Step function F for IVC: `z_{e+1} = F(z_e, π_e, pub_e)` where z = (epochs, violations≡0, drawdown-bucket max, volume accumulator, mandate-class hash chain). Folding verifies π_e's public inputs chain correctly (state roots consecutive, nullifier fresh) and updates z. Terminal proof: "there exists a valid IVC chain of length N from genesis z_0 to z_N" — one Honk proof, ~2KB, verified by anyone. Soundness inherits the folding scheme's knowledge soundness; the credential cannot be minted without actually having produced every epoch proof.

F.6 Security model (who can do what)

Adversary	Capability	Worst case	Mitigation
Malicious agent	Arbitrary strategy, tries to breach mandate	Nothing: breaching intents are unprovable → excluded	Circuit soundness (primary invariant)
Malicious agent + stale proof replay	Reuse old proof	Rejected	Epoch + state-root binding, nullifiers
Colluding keypers (≥t)	Decrypt epoch flow early	Front-run/copy that epoch's flow — **privacy breach, not theft**	Slashing + traitor tracing; funds untouched; batches still clear at uniform price so even leaked flow is hard to sandwich

Adversary	Capability	Worst case	Mitigation
Auction operator (MVP centralization)	Censor/withhold	Liveness delay; challenged via DA + watchtowers; forced-inclusion via L1	Escape hatch; decentralize in research version
TEE break (fast lane users)	Read enclave strategy	Strategy leak for opted-in agents only	P2: never integrity-bearing
Statistical observer	Watch public transcripts over months	Coarse inference (aggregate flow, bucketed reputation)	Buckets, padding/decoys (§F.7), acknowledged residual (§K)
Malicious principal	Frame own agent?	Can only revoke/withdraw own funds	Roles separated by construction

F.7 Privacy guarantees and their edges

Guaranteed (cryptographic): pre-trade order confidentiality (< t keypers); portfolio confidentiality (Poseidon hiding commitments); strategy code never touches the protocol; per-trade unlinkability within batches; credential reveals only declared buckets (zk). **Not** guaranteed (stated honestly): aggregate market-level flow is visible (clearing prices/volumes are public — necessarily, they're the market); timing/participation metadata (an agent's *presence* in epochs can be padded with decoy ciphertexts at a gas cost, a knob not a default); long-horizon statistical inference against a persistent pseudonym (the price of accountability — mitigable by identity rotation at the cost of reputation reset, a tradeoff the agent chooses).

F.8 Known attack vectors ranked by severity

1. **Circuit bugs** (soundness break = silent mandate bypass). Mitigation: minimal circuits, formal verification of the policy circuit (small enough to be tractable), redundant account-level plaintext caps (E.10-2), audits, bounty.
2. **Oracle manipulation** (prices feed leverage/drawdown checks). Mitigation: median of signed feeds, staleness bounds in-circuit, breaker bit.
3. **Keyper collusion** (privacy). Above.
4. **Reputation gaming via self-dealing** (wash-trade a good record). Hard but not impossible in uniform-price auctions (you mostly trade against yourself at fair price and pay fees); volume buckets discount low-adversarial-cost metrics; performance buckets are net-of-fee. Named as an open research item (§K).
5. **Griefing via challenge spam** — bonded challenges.
6. **DA withholding** — challenge → pause → L1 exit.

F.9 Limitations (cryptographic)

No post-quantum security in the core (pairings, ECDSA) — STARK-based migration path named, not built. Trusted-setup residue (universal, but present). Proof latency floors out around ~10^2ms — sub-batch-interval by design, but permanently incompatible with microsecond trading. FHE-style *encrypted matching* (auction computed without any decryption ever) is excluded as impractical in 2026 and scoped to §J research track.

G. Agentic Trading Flow (end-to-end)

0. Setup (principal). Deploy Mandate Account; deposit USDC; define P = {whitelist: {ETH, WBTC, SOL-wormhole}, max_position: $100k/asset, max_leverage: 2×, max_daily_drawdown: 3%, max_orders_per_epoch: 5, expiry: 90d}; publish `C_P = Poseidon(P)`; hand P (plaintext) + a session key to the agent operator; keep the viewing key.

1. Agent registration. Operator generates `(pk_agent, salt)`, registers `id = Poseidon(pk_agent, salt)` and the session-key authorization in the registry; genesis reputation state z_0 committed. Optional: post a TEE attestation quote to earn a "hardware-shielded" tag (cosmetic tier, not trust-bearing).

2. Prove authorization (per action, implicit). Every user-operation carries the session signature; the account contract checks scope/expiry/revocation on-chain. Authorization is thus continuously proven, not a one-time event.

3. Strategy runs (off-chain, dark). The agent — an LLM planner, an RL policy, a plain momentum script; the protocol is agnostic — decides: buy 3 ETH, limit 3,120 in epoch e.

4. Prove compliance. Agent fetches current `R_e` + its Merkle paths (from its own state cache, verified against the on-chain root), runs the Noir policy circuit → π_e in ~300ms.

5. Encrypt & submit. `ct = IBE.Enc(epoch_e, order)`; submit `(ct, com(o), π_e, session_sig)` to the intent lane; it lands in the epoch-e blob before cutoff. On-chain verifier checks π_e (aggregated); invalid proofs never enter the auction.

6. Batch clears. Epoch closes → keypers release `sk_e` → orders decrypt → uniform $p^* = 3{,}118$ computed → agent filled 3 ETH @ 3,118 alongside all other epoch flow at the same price. No one front-ran it; no one can attribute the fill.

7. Cross-chain leg (if any). Had the intent targeted, say, a Base-native asset, the cleared residual becomes an ERC-7683 order; a solver checks the agent's standing proof, fills on Base, settles through the rail's escrow.

8. State updates. Settlement contract updates the commitment leaves (with the fill-delta proof), producing `R_{e+1}`. Watchtower window opens.

9. Risk management, continuously. Next epoch's proof must verify against `R_{e+1}` — drawdown and leverage checks therefore reflect *actual settled reality*, not the agent's claims. If the day's losses hit 3%, tomorrow's loss-increasing orders are simply unprovable; the agent can only de-risk. If the principal panics: one revoke transaction, effective next epoch, worst-case exposure = one epoch inside mandate.

10. Reputation accrues. π_e folds into the running IVC instance. After six months: `z = (N=13,824 epochs, violations=0, max_dd_bucket=B2(<5%), volume_bucket=V4, mandate_class=conservative-spot)`. Terminal proof posted monthly. A vault on an entirely different chain verifies this one proof and programmatically allocates $250k under a fresh mandate. The agent never revealed one trade.

H. Comparison

vs.	Their frame	MANDATE's difference
Ethereum	Transparent global settlement	MANDATE is Ethereum-aligned middleware; it *adds* the confidentiality + verifiable-delegation layer Ethereum L1 will never have natively (even enshrined encrypted mempools cover transit only)
Solana	Win by latency	MANDATE concedes the microsecond game and changes the payoff matrix: in uniform-price batches, speed buys nothing. Solana agents leak everything to the leader; MANDATE agents leak nothing to anyone
Avalanche / Cosmos app-chains	Sovereign chain per app (dYdX v4 pattern)	Sovereignty fragments liquidity and still runs transparent books. MANDATE keeps Ethereum liquidity and adds privacy the app-chain lacks
Aztec	General-purpose private smart contracts	Complementary, not competitive: Aztec is a *platform*, MANDATE is a *market + trust structure*. Aztec (2026) gives no batch auctions, no anchored-state policy proofs, no folding reputation. Plausible future: MANDATE-on-Aztec as the execution venue (§L)
Aleo	Private L1, off-chain execution	Same privacy instinct, fatal liquidity island; MANDATE goes where the money already is
Celestia	DA commodity	Orthogonal; a supplier
EigenLayer	Security commodity	Orthogonal; a supplier (keypers/watchtowers as AVS)
CoW / UniswapX / 7683 intent stack	Best execution for *visible* intents	MANDATE = "7683 with a trust layer": adds *who may sign* (mandates) and *how they've behaved* (credentials) to a stack that only standardized *what is signed*. Fully interoperable — MANDATE emits 7683 orders
Anoma	Intents as first-class L1 primitive	Grand unified theory vs. shipping wedge: MANDATE is three contracts and two circuits on existing rails, and its mandate/reputation layer is absent from Anoma's design regardless
Cobo-style agentic wallets	Plaintext policy enforcement, operator-trusted	The most instructive contrast: identical *goal* (safe agent autonomy), inverted *trust*: their guarantees are legible only to the customer and rest on the operator; MANDATE's are legible to the world and rest on math. They validate the market; MANDATE removes their trust assumption

---

I. MVP — "MANDATE Lite" (student-buildable, 4 weeks)

Scope discipline: one L2 testnet (Base Sepolia), one market (mock-ETH/mock-USDC), 10-second epochs, commit–reveal *simulating* threshold encryption, an on-chain reputation counter *simulating* folding, no cross-chain, no TEE. The zk policy proof — the genuinely novel part — is real, not simulated. That's the correct place to spend your credibility.

I.1 Core user flow (demo script)

1. Judge/principal opens the dashboard, creates a Mandate Account, sets sliders (whitelist, max notional, daily loss cap), deposits testnet tokens → sees `C_P` on-chain.
2. Starts two bundled agents (momentum + market-maker). Watches encrypted intents (opaque blobs) arrive per epoch, proofs verify, batches clear at a uniform price, portfolio updates via commitments.
3. Presses "make agent misbehave" → the agent *tries* an over-limit order → prover fails → order never exists on-chain. This 10-second moment is the whole pitch.
4. Opens the reputation tab: epochs survived, zero violations, drawdown bucket — then revokes the agent live and shows funds intact.

I.2 Tech stack

- **Circuits:** Noir + Barretenberg (bb.js for browser/Node proving). One circuit: `policy_check` (whitelist Merkle proof, notional cap, position cap, daily-loss cap vs. HWM leaf, epoch binding). Target < 50k constraints.
- **Contracts (Solidity, Foundry):** `MandateAccount` (minimal 4337-style: owner key + session key + verifier gate — you can shortcut full EntryPoint integration with a custom account), `MandateRegistry`, `BatchAuction` (commit ct-hash in epoch e, reveal in e+1, uniform-price clearing for one pair, naive O(n log n) on ≤ 50 orders is fine), `StateTree` (Poseidon incremental Merkle tree; use a maintained IMT library), `HonkVerifier` (autogenerated by bb), `Reputation` (per-agent counters updated only on verified epochs — the honest simulation of folding).
- **Backend (TypeScript, Node + viem):** sequencer service — collect submissions, enforce cutoff, post batch, reveal, trigger clearing; a mock price oracle (signed feed); a small indexer (SQLite) for the UI.
- **Agents (TypeScript or Python):** tiny framework: `Strategy` interface → `decide(marketState) -> Order[]`; pipeline `decide → prove (bb) → encrypt(commit) → submit`. Ship the two demo strategies. This is also your SDK seed.
- **Frontend (Next.js + wagmi):** principal dashboard + a "protocol explorer" view showing ciphertext-in / uniform-price-out, which makes the privacy visible.

I.3 Simulate vs. build (and say so in the README)

Component	MVP treatment	Honest label
Policy zk proof	**Real** (Noir/Honk, verified on-chain)	The contribution
Threshold encryption	Commit–reveal by a single sequencer	"Ring 1 simulated; slot-in Shutter-style IBE is engineering, not research"
Folding reputation	Verified-epoch counters on-chain	"Constant-size credential deferred; counters preserve the semantics"
State-transition proof	Plain contract execution over commitments	Ring 2 preserved; operator-proving deferred
Cross-chain (7683)	Omitted or stubbed as an event	Roadmap
Watchtowers/AVS	Omitted	Roadmap

I.4 Four-week roadmap

- **W1 — the circuit.** Noir `policy_check` + unit tests (positive/negative vectors for every constraint); generate Solidity verifier; prove from Node. Milestone: on-chain-verified proof of a compliant order.
- **W2 — market structure.** BatchAuction commit–reveal + clearing; StateTree; MandateAccount gating on proof + session key. Milestone: two hardcoded orders clear at uniform price, commitments update.
- **W3 — agents + service.** Sequencer service, oracle, agent framework, momentum + MM strategies, negative-path demo (unprovable order). Milestone: unattended 30-minute run, 180 epochs, zero manual steps.
- **W4 — surface.** Dashboard, explorer view, reputation tab, revocation flow, README/architecture doc (this document, trimmed), 3-minute demo video. Milestone: a stranger can run `docker compose up` and reproduce the demo.

Deliberately out of scope even if tempted: multi-market, real price feeds, gas golf, token anything.

J. Research-Grade Version (serious team, 12–24 months)

Phase 1 (months 1–6) — cryptographic core. Formally specify + verify the policy and state-transition circuits (small enough for tools like Lean/EasyCrypt-adjacent circuit verification to bite); production Noir circuits with lookup-heavy range checks; folding-based reputation (Nova-lineage over a curve cycle, Honk wrap) with a written knowledge-soundness argument for the composed system; integrate a real threshold-IBE lane (partner with or fork the Shutter/Primev stack rather than rebuilding keypers); publish the protocol paper.

Phase 2 (months 6–12) — decentralize the operator. Keypers and watchtowers as restaked AVS operator sets with concrete, objectively-attributable slashing (premature-decryption traitor tracing; state-root fraud proofs); proved off-chain auction execution (the operator posts a validity proof of clearing, removing the "honest auction" assumption); DA to blobs with reconstruction tooling; forced-inclusion path from L1; sub-2s epochs with proof aggregation (shared aggregation prover amortizing verification to O(1) per agent).

Phase 3 (months 12–24) — market depth + interop. ERC-7683 adapter in production with standing-proof verification for solvers; multi-market auctions with cross-margining inside the commitment tree (leverage circuit over a portfolio vector — the hard circuit); reputation credential standard (aim: an ERC) so third-party vaults/protocols consume credentials natively; selective-disclosure tooling for allocators/auditors (BBS+ attestations from viewing-key holders); optional TEE fast-lane tier with attestation verified on-chain; economic red-team of the auction (collusion, decoy-order costs, wash-trading economics) with published parameters. Research track kept explicitly separate from the roadmap: FHE/MPC *encrypted matching* (clearing without any decryption event, killing the keyper assumption entirely) — revisit when threshold-FHE matching over ~10^3 orders drops under ~1s.

Team shape: 2 zk engineers, 1 cryptographer, 2 protocol/Solidity, 1 market-microstructure researcher, 1 devrel. That's a seed-stage protocol company, not a moonshot lab.

K. Risks — the brutal section

Possibly fatal, named first.

1. **The cold-start liquidity problem is the real boss fight, not cryptography.** Batch auctions need flow to price well; agents need pricing to bring flow. Penumbra's shielded DEX — technically excellent — demonstrates that privacy tech does not conjure liquidity. Mitigations (solver-bridged external liquidity so early batches clear against AMM quotes; a niche beachhead like private treasury rebalancing) are plausible, not proven. If this fails, everything above is a beautiful paper.
2. **Reputation may not be the commodity I claim.** The thesis assumes allocators will pay for *verified constraint-compliance history*. They might instead keep allocating on brand, audited track records, and legal recourse. Zero-knowledge credentials compete with "I know the founder." This is a market hypothesis, not a theorem.
3. **Statistical privacy erosion is permanent and cumulative.** Clearing prices, volumes, participation timing, and bucketed credentials leak bits every epoch, forever. Against a motivated quant adversary with years of transcripts, "strategy privacy" degrades to "strategy obscurity." I can price decoys and pad timing; I cannot repeal statistics. Anyone promising absolute strategy privacy on a public settlement layer is selling something.

Expensive or hard.

4. Wash-traded reputation: an attacker funding both sides of uniform-price batches pays only fees and spread to mint "volume + zero violations." Fee floors and net-of-fee performance buckets raise the cost; they don't zero it. Open problem, flagged in the credential semantics ("volume" must never be read as "skill").
5. Circuit soundness is a single point of catastrophic failure — one bug converts "provably safe agent" into "provably-labeled unsafe agent," which is worse than no label. Hence formal verification + redundant plaintext caps, and honest acknowledgment that verified circuits are a research-grade cost.
6. Oracles: every leverage/drawdown proof is downstream of price feeds. Standard problem, non-negotiable dependency.
7. Proof latency confines the design to batch-frequency (seconds) trading. This is embraced, not solved — but it also caps the addressable strategy universe (no latency arb, no microstructure alpha). Some of the most lucrative agent strategies are excluded *by construction*.

Only theoretical / hard to prove.

8. Composed-system soundness (folding + Honk wrap + circuit chain + contract logic) has more interfaces than any single paper covers; the security argument for the *composition* is real work, not citation.
9. Silent-setup threshold encryption and traitor tracing are young literature — deployed versions may differ from the papers relied on.

Copyable.

10. Everything here is contracts + circuits on public rails: Aztec could ship a native version with better privacy plumbing; CoW could bolt mandates onto its auctions; a Cobo could publish proofs of its Pacts. The defensible assets are (a) the reputation-credential network effect — history is non-portable to a competitor by definition — and (b) circuit/standard mindshare (become the ERC). Eighteen-month head start, not a moat.

Regulatory.

11. This is, functionally, dark-pool infrastructure for autonomous traders. Expect scrutiny on: pooled delegation resembling collective investment schemes (managed-fund / investment-adviser perimeters in most jurisdictions), sanctions exposure of privacy pools (the Tornado Cash lineage of enforcement), market-abuse rules that bind regardless of agency ("my bot did it" is not a defense — and MANDATE's proofs may actually *help* here, since a mandate is documented, enforced supervision), and MiCA/CFTC-style algorithmic-trading registration regimes that keep evolving through 2026. The selective-disclosure viewing-key design is the compliance story: *auditable by consent, opaque by default*. Whether regulators accept that inversion is undetermined and jurisdiction-dependent. Build the disclosure tooling from day one; do not market anonymity.

L. Final Recommendation

Build it as a cryptographic middleware + app-specific settlement layer on Ethereum rollups — emphatically not a new chain, and not (yet) its own rollup.

Reasoning, compressed: the value proposition is a *trust structure* (mandates, proofs, credentials) wrapped around a *market structure* (encrypted batch auctions). Neither needs sovereign blockspace; both need adjacency to existing liquidity and composability with the 4337/7683/restaking stack — which is only available *on* Ethereum's rails. A new L1 maximizes the cold-start problem (risk #1) for zero architectural gain; a dedicated rollup becomes attractive only at Phase-2 scale, when proved off-chain auction execution and sub-2s epochs justify owning the sequencing pipeline — and by then the contracts migrate into it cleanly. If Aztec's execution layer matures on schedule, a MANDATE-on-Aztec deployment is the natural privacy-maximalist venue; the middleware framing keeps that door open instead of competing with it.

Sequencing for you specifically: **(1)** build MANDATE Lite exactly as scoped in §I — the four-week artifact is a hackathon winner and a working proof of the only novel claim; **(2)** write the whitepaper from §§D–G with §K intact (the risk section is what makes researchers take it seriously); **(3)** publish the policy circuit + credential format as a draft standard early — in this design, the standard *is* the moat.