TECHNICAL WHITE PAPER

Blockchain Engineering & Agentic AI Infrastructure

Maticz Technologies

Document Type	Engineering Specification — Architecture, Standards & Protocol Reference
Version	Revision Date: Q1 2026
Scope	Blockchain Layer 1 & 2, ZK Systems, DeFi Protocols, Tokenization Standards, Agentic AI, MEV Infrastructure
Classification	Confidential — For Technical Review, Integration Partners & Institutional Clients

Table Of Content

1. Introduction & Scope

This document is a formal technical specification for the full-stack blockchain and agentic AI engineering services offered by **Maticz Technologies**. It is written for software architects, protocol engineers, smart contract auditors, and institutional technology partners who require precise detail on the standards, algorithms, network parameters, and architectural patterns underpinning every service layer.

The following sections are organized by functional domain. Each section presents:

(1) The standards and protocols adopted

(2) The engineering architecture and design rationale

(3) Measurable performance targets and security requirements

(4) Integration interfaces. Marketing language has been deliberately excluded; all claims are accompanied by the underlying technical reference.

2. Agentic AI Infrastructure

2.1 Agent Architecture

Agentic AI systems differ from chatbots in that they operate on extended task horizons, maintain state across multiple reasoning steps, call external tools, and take irreversible actions (on-chain transactions, API calls). Our agent architecture is built on three interacting layers:

Parameter	Specification / Standard
Perception Layer	Retrieves context: on-chain state via RPC, off-chain data via RAG, real-time feeds via WebSocket subscriptions (DEX events, oracle updates)
Reasoning Layer	LLM inference: structured Chain-of-Thought (CoT) with scratchpad; tool-use via OpenAI Function Calling / Anthropic Tool Use spec; ReAct loop (Reason → Act → Observe → repeat)
Action Layer	Executes tool calls: sign & broadcast transactions, call REST APIs, write to databases, spawn sub-agents. All actions logged to append-only audit trail.

2.2 LLM Integration

- **Model serving:** vLLM (PagedAttention memory management) for GPU-efficient inference of open-weight models (LLaMA-3 70B, Mistral, Mixtral-8x22B). Supports continuous batching for throughput optimization.

- **Quantization:** GPTQ (4-bit, 8-bit) or AWQ (Activation-aware Weight Quantization) for inference memory reduction without significant accuracy loss. llama.cpp for CPU-only edge deployments.

- **Context window:** Extended context via RoPE scaling (YaRN) to support 128k+ token windows needed for large codebase or full protocol documentation ingestion.

- **Fine-tuning:** LoRA (Low-Rank Adaptation) and QLoRA (quantized LoRA) for domain-adapted models trained on DeFi protocol documentation, Solidity code, and on-chain transaction data.

2.3 Retrieval-Augmented Generation (RAG) Pipeline

To ground agent reasoning in accurate, up-to-date knowledge, we implement an Agentic RAG pipeline with the following components:

- **Document ingestion:** PDFs, Markdown, on-chain ABI JSONs, subgraph schemas, and governance forum posts are chunked (512 token chunks with 64-token overlap using recursive character text splitter) and embedded.
- **Embedding model:** text-embedding-3-large (OpenAI, 3072 dimensions) or E5-large-v2 (open-weight, MTEB benchmark top-5). Embeddings stored in Qdrant (vector database with HNSW indexing: ef_construction=200, m=16, providing ~99% recall@10 at millisecond latency).
- **Retrieval:** Hybrid search — dense vector similarity (cosine distance) + sparse BM25 keyword scoring, fused via Reciprocal Rank Fusion (RRF). Top-k=20 candidates retrieved; re-ranked via cross-encoder (ms-marco-MiniLM-L-12-v2) to top-k=5 for context window injection.
- **On-chain RAG:** Real-time smart contract state is fetched via eth_call (view functions) and injected as structured context. Subgraph (The Graph Protocol) queries provide aggregated historical protocol metrics (total value locked, volume, liquidations).

Parameter	Specification / Standard
Embedding dimensions	3072 (text-embedding-3-large) / 1024 (E5-large-v2)
Vector index	HNSW (Hierarchical Navigable Small World); ef_construction=200, m=16
Retrieval latency (p99)	< 50 ms for top-20 candidates from 10M document corpus
Re-ranking model	Cross-encoder: ms-marco-MiniLM-L-12-v2 (ONNX runtime, 2 ms/query)
Chunking strategy	Recursive character splitter; 512 token chunks; 64 token overlap

2.4 Agent Orchestration Frameworks

We implement agent orchestration using LangGraph (stateful multi-agent graphs with cycles) as the primary framework, with the following topology:

- **Graph nodes:** Each node is a Python function or LLM call. Edges are conditional (based on agent output) or unconditional. State is a TypedDict passed between nodes and persisted in a Redis-backed checkpoint store.
- **Tool registry:** Tools are defined as Pydantic BaseModel schemas; the LLM selects tools via structured output (JSON-mode with schema enforcement). Tool execution is isolated in sandboxed subprocess workers with timeout and resource limits.
- **Multi-agent patterns:** Supervisor agent routes subtasks to specialist sub-agents (DeFi analyst, contract deployer, risk assessor). Sub-agents operate with restricted tool access (principle of least authority).
- **Memory types:** In-context (current conversation window), external (vector store long-term memory via MemGPT-style paging), structured (relational DB for user preferences and position history), and episodic (summarized past interactions injected as system context).

2.5 On-Chain Agent Security

Autonomous agents that sign and broadcast on-chain transactions require strict security controls:

- **Key management:** Agent signing keys are generated and stored in AWS KMS (FIPS 140-2 Level 3 HSM) or HashiCorp Vault (Transit secrets engine). Private keys never exist in plaintext in memory. All sign requests are audited.
- **Transaction simulation:** Every transaction is simulated via eth_call (Ethereum) or simulateTransaction (Solana) before broadcasting. Simulation checks: expected token balance changes, gas estimation within bounds, no revert condition.
- **Spending limits:** On-chain guardian contracts enforce per-transaction and daily spending limits. Transactions exceeding limits require a multi-sig approval (Gnosis Safe with 2/3 or 3/5 signers).
- **Pause mechanism:** Agents subscribe to protocol pause events (circuit breakers). If a protocol emits a Paused event, all pending agent actions for that protocol are queued pending human review.

3. Crypto Exchange Ecosystem

3.1 Centralized Exchange (CEX) Architecture

Our White Label CEX stack is engineered for institutional-grade throughput and compliance. The core components and their technical specifications are:

Parameter	Specification / Standard
Matching engine	Price-time-priority (PTP) CLOB; in-memory order book (red-black tree); target throughput > 1M orders/sec; written in Rust or C++ for deterministic latency
Order types	Limit, Market, Stop-Limit, Stop-Market, Trailing Stop, Post-Only, Fill-or-Kill (FOK), Immediate-or-Cancel (IOC)
Settlement	T+0 internal ledger settlement; net settlement across user accounts with atomic double-entry bookkeeping
KYC/AML module	Onfido / Sumsub SDK integration; tiered KYC levels (L1: email + phone; L2: ID document + liveness; L3: enhanced due diligence for institutional). OFAC screening on every deposit/withdrawal via Chainalysis API
Custody	Cold/warm/hot wallet split: 95% cold (HSM-backed, multi-sig), 4% warm (time-locked multi-sig), 1% hot (auto-funded, per-withdrawal limit enforced)
API interfaces	REST (HTTPS, JSON), WebSocket (real-time order book / trade feed), FIX 4.4 protocol for institutional clients (binary, low-latency)
Multi-currency	ERC-20 deposit tracking via event log indexing (Transfer event, keccak256 topic filter); UTXO chain deposits via xPub-derived address generation (BIP-32/44/84)

3.2 Decentralized Exchange (DEX) — AMM Engineering

Our AMM implementations are built on well-defined invariant functions. The choice of invariant determines the pricing curve and capital efficiency:

Constant Product AMM (Uniswap v2 Model)

Invariant: $x * y = k$ where x, y are reserve quantities. Spot price: $P = y / x$. Price impact for trade Δx: $\Delta y = (y * \Delta x) / (x + \Delta x)$. Fee: Applied as a reduction on input (e.g., 0.3% → multiply input by 0.997 before computing output). Liquidity is uniformly distributed across [0, ∞), resulting in low capital efficiency for assets with narrow price ranges.

Concentrated Liquidity AMM (Uniswap v3 Model)

Liquidity providers specify a price range [Pa, Pb]. Within this range, the effective invariant is: $(x + L/\sqrt{Pb})(y + L*\sqrt{Pa}) = L^2$. Capital efficiency scales as $\sqrt{(Pb/Pa)}$ relative to full-range liquidity. Tick spacing is configurable (1, 10, 60, or 200 ticks) based on fee tier (0.01%, 0.05%, 0.3%, or 1%).

Positions are represented as ERC-721 NFTs storing: tickLower, tickUpper, liquidity, feeGrowthInside0LastX128, feeGrowthInside1LastX128.

- **TWAP oracle:** Each pool stores a cumulative sum of tick * time (tick accumulator). TWAP over a window [t0, t1]: avgTick = (acc[t1] - acc[t0]) / (t1 - t0). Resistant to single-block manipulation; manipulation cost scales linearly with window length and pool liquidity.

StableSwap Invariant (Curve Model)

For stable assets, the StableSwap invariant blends constant-sum (x + y = k, zero slippage) with constant-product: A * n^n * Σxi + D = A * D * n^n + D^(n+1) / (n^n * Πxi) where A is the amplification coefficient (tunable governance parameter), D is the invariant, and n is the number of assets. At low A, the curve approaches constant-product; at high A, it approaches constant-sum, minimizing slippage for pegged assets.

3.3 P2P Exchange & Escrow Architecture

Peer-to-peer trading platforms require trustless escrow without a central counterparty. Our smart contract escrow pattern:

- **Escrow contract:** Seller locks tokens via ERC-20 approve + transferFrom into the escrow contract. Release condition: buyer confirms receipt off-chain; arbitration path available if disputed.

- **Dispute resolution:** Trusted arbiter (multi-sig or DAO) can release funds to either party. Arbiter selection uses a reputation-weighted random selection from a staked arbiter registry.

- **Encrypted communications:** Trade-specific messaging uses ECIES (Elliptic Curve Integrated Encryption Scheme) with ephemeral keys derived from both parties' public keys. Messages stored off-chain (IPFS or encrypted DB) with on-chain hash commitment.

3.4 Derivatives & Margin Trading

Complex financial instruments require precise margin accounting and liquidation mechanics:

Parameter	Specification / Standard
Instrument types	Perpetual futures (funding rate mechanism), dated futures, options (European/American), variance swaps
Funding rate	Perpetuals: funding = clamp(premium / 24h, -0.05%, 0.05%) paid every 8 h between longs and shorts; keeps perp price anchored to index
Margin model	Cross-margin (shared margin pool across positions) and isolated margin (per-position risk limit). Initial margin and maintenance margin defined per-asset by risk parameters

Liquidation engine	Background keeper bots monitor Health Factor continuously; partial liquidation preferred; insurance fund absorbs bad debt beyond collateral value
Copy trading	Signal provider's position deltas replicated to follower accounts proportionally to their allocated capital; slippage budget enforced per replication trade

3.5 Algorithmic Trading Bot Infrastructure

Maximal Extractable Value (MEV) is the additional value extractable by controlling transaction ordering within a block. Our MEV tooling addresses both MEV capture and MEV protection:

- **Ethereum MEV:** Validators running MEV-Boost select the highest-paying block from a competitive market of block builders (Flashbots, BloXroute). Our bots submit bundles via eth_sendBundle API (signed JSON payload: {txs, blockNumber, minTimestamp, maxTimestamp, revertingTxHashes}). Bundle atomicity ensures all-or-nothing execution.

- **Solana MEV (Jito):** Modified Solana validator accepts off-chain bundles via Jito Block Engine. Bundles of up to 5 transactions land atomically. Geyser plugin provides sub-10 ms latency notification from transaction landing to searcher.

Parameter	Specification / Standard
Strategy type	Cross-exchange triangular arbitrage, CEX-DEX arbitrage, flash-loan-enabled same-block arbitrage
Latency requirement	< 50 ms for opportunity detection + transaction submission (co-location or dedicated nodes)
Flash loan provider	Aave v3 (ERC-3156 compliant), Balancer (no-fee flash loans for within-batch operations)
Profit calculation	Simulate full execution path via Tenderly simulation API before submission; abort if net profit < gas cost + bundle tip
Gas strategy	EIP-1559: maxFeePerGas = baseFee * 1.2 + priorityFee; priorityFee dynamically adjusted based on mempool competitiveness

4. Prediction Market Protocol Engineering

4.1 Market Architecture

Prediction markets allow participants to trade binary or scalar outcomes. The on-chain architecture consists of:

- **Market factory:** Deploys individual market contracts for each question. Each market has: resolution timestamp, oracle address, category, outcome token pair.

- **Outcome tokens:** Conditional tokens (ERC-1155, Gnosis CTF standard) representing YES / NO positions. Total supply of YES + NO always equals total liquidity deposited. Prices sum to 1 USDC (implied probability interpretation).

- **AMM pricing:** LMSR (Logarithmic Market Scoring Rule) provides guaranteed liquidity at any trade size; market maker absorbs risk in exchange for fees. Cost function: $C(q) = b * \ln(\Sigma \exp(qi/b))$ where b is the liquidity parameter and qi is the quantity of outcome i. Alternatively, Uniswap-style CPMMs are used for simpler markets.

- **Resolution:** Chainlink oracle reports binary outcome (0 or 1) at resolution time. If no oracle is available, UMA Optimistic Oracle is used: any address can propose a resolution with a bond; disputed resolutions go to UMA's DVM (Data Verification Mechanism) for token-holder vote.

4.2 Oracle Security for Resolution

The security of prediction markets depends entirely on tamper-proof oracle resolution. Our multi-layer oracle security approach:

- **Primary:** Chainlink Any API (custom external adapter) for verifiable event data (sports scores via Sportradar, election results via AP/Reuters, financial data via Refinitiv).

- **Fallback:** UMA Optimistic Oracle with 72-hour dispute window and economic bond (proposer stakes USDC; incorrect resolution means bond is slashed and distributed to disputers).

- **Escalation:** Disputed markets escalate to a Kleros arbitration court (decentralized juror selection via sortition, jurors stake PNK token, ruled by Schelling point incentives).

---

5. RWA Tokenization & Stablecoin Architecture

5.1 Real-World Asset (RWA) Tokenization

Tokenizing real-world assets (real estate, commodities, private credit) requires a legal and technical bridge between off-chain legal ownership and on-chain token representation:

- **Legal wrapper:** Special Purpose Vehicle (SPV) or trust holds the underlying asset. Token represents a beneficial interest in the SPV. Legal enforceability is jurisdiction-specific (Delaware LLC, Cayman SPC, or Singapore VCC structures most common).

- **Token standard:** ERC-3643 (T-REX — Token for Regulated Exchanges). Built on ERC-20 with an Identity Registry (on-chain KYC/AML whitelist using ERC-735 claims), Transfer Rules module (transfer restrictions enforced in _beforeTokenTransfer hook), and Compliance module (hold periods, investor limits).

- **Oracle integration:** Chainlink Proof of Reserve for on-chain attestation of off-chain asset value. NAV (Net Asset Value) updates published on-chain at configurable intervals (daily for liquid assets, weekly for illiquid).

- **Custody:** Institutional-grade custody partners (Fireblocks, Copper) hold underlying assets; API webhooks trigger on-chain events when off-chain events occur (asset transfer, valuation update).

5.2 Stablecoin Architecture

Parameter	Specification / Standard
Fiat-backed (e.g., USDC model)	1:1 backing in segregated bank accounts or T-bills. Monthly reserve attestation by Big-4 auditor. Redemption: ERC-20 burn on-chain → fiat wire off-chain (T+1 settlement).
Crypto-collateralized (e.g., MakerDAO model)	Over-collateralized (e.g., 150% ETH backing for 100 DAI minted). Stability fee (borrow rate) and liquidation ratio governed by MKR token votes. Keeper bots liquidate undercollateralized CDPs.
Algorithmic (hybrid)	Partial collateral + seigniorage mechanism. Redemption arbitrage maintains peg: if stablecoin < $1, burn stablecoin for $1 of governance token; if > $1, mint stablecoin and sell. Vulnerable to bank-run death spirals (see Terra/Luna post-mortem — mandatory reading for design decisions).
Yield-bearing	ERC-4626 vault wrapping fiat-backed stablecoin; underlying deposited in T-bill money-market protocol (Ondo Finance / Superstate model). Share price appreciates vs fixed NAV.

6. Blockchain Architecture & Layer Strategy

6.1 Layer 1 Network Integration

Maticz builds directly on production Layer 1 networks. The primary chains and their relevant technical characteristics that govern our engineering decisions are as follows.

Parameter	Specification / Standard
Ethereum (EVM)	Proof-of-Stake (Gasper: Casper FFG + LMD-GHOST). Block time: ~12 s. Finality: ~15 min (2-epoch). EIP-1559 base-fee model. State

	trie: Modified Merkle-Patricia Trie (MPT). Execution: EVM bytecode (256-bit word size).
Solana (SVM)	Proof-of-History (PoH) + Tower BFT. Block time: ~400 ms. Finality: ~1.3 s (optimistic). Turbine block propagation. Runtime: BPF (Berkeley Packet Filter) bytecode. Parallel execution via Sealevel; account model with explicit state separation.
BNB Smart Chain	Proof-of-Staked-Authority (PoSA). 21-validator set. Block time: ~3 s. EVM-compatible. Consensus messages via LibP2P gossip. Cross-chain communication via BNB Bridge (IBC-inspired).
Avalanche (C-Chain)	Snowman consensus (linear chain variant of Avalanche protocol). Sub-second probabilistic finality. EVM-compatible C-Chain with coreth client. P-Chain handles validator staking; X-Chain for asset exchange (DAG-based UTXO).

Our node infrastructure runs dedicated archive nodes for Ethereum (Erigon client for efficient flat-file state storage), Solana (Jito-Solana validator client for MEV-aware execution), and Avalanche (AvalancheGo). All nodes expose authenticated JSON-RPC 2.0 endpoints (eth_* and web3_*) over TLS 1.3 with mutual certificate authentication.

6.2 Layer 2 Rollup Architecture

Scaling to production transaction throughput requires off-chain execution with on-chain data availability. We deploy and operate within two classes of rollup:

6.2.1 Optimistic Rollups (ORU)

Implemented via Arbitrum One (Nitro stack) and Optimism (OP Stack). The fraud-proof window is 7 days (Ethereum block-time dependent). Withdrawal finality under the canonical bridge is therefore 7 days; fast-withdrawal services (e.g., Hop Protocol, Across) reduce this to minutes at the cost of a liquidity provider fee.

- **Arbitrum Nitro:** WASM-based fraud proofs (Bisection game). AVM legacy deprecated. Stylus extension allows WASM contracts compiled from Rust, C, or C++.
- **OP Stack:** Cannon fault-proof system (MIPS-based). Multi-proof roadmap targets multiple independent verifiers. The Superchain architecture enables shared sequencer & bridge across chains (Base, Mode, Zora, etc.).
- **Gas model:** L2 execution gas is priced against L2 base-fee; L1 data posting cost (calldata or blob) is passed through as a separate fee component.

With EIP-4844 (Proto-Danksharding) active on Ethereum, we configure sequencers to publish transaction batches as blob-carrying transactions (blob type 0x03). Blobs are 128 KB each,

priced via a separate blob base-fee market, and retained on beacon nodes for ~18 days. This reduces L1 data cost by 80–95% compared to calldata.

Parameter	Specification / Standard
Blob size	131,072 bytes (128 KiB) per blob; max 6 blobs per block (EIP-4844 mainnet parameter)
Blob fee market	EIP-4844 excess_blob_gas exponential pricing; target: 3 blobs/block
ORU finality (canonical)	~7 days (fraud-proof challenge window)
ORU finality (fast-exit)	Minutes via liquidity-provider bridges
Arbitrum throughput	Up to ~40,000 TPS (theoretical Nitro); practical ~4,000 TPS sustained

6.2.2 ZK Rollups — zk-SNARKs & zk-STARKs

Zero-Knowledge proof systems provide cryptographic finality without a challenge window, reducing withdrawal time to minutes (proof generation + L1 verification). We deploy two proof system families depending on security requirements and circuit complexity:

zk-SNARKs (Succinct Non-Interactive Arguments of Knowledge)

- **Proof system:** Groth16 (trusted setup, smallest proof size: ~200 bytes, fastest on-chain verification ~300k gas) or PLONK / UltraPLONK (universal trusted setup via Powers-of-Tau ceremony; supports recursive proofs).
- **Elliptic curve:** BN254 (also called alt_bn128) for Ethereum-native pairing precompiles (EIP-196, EIP-197); BLS12-381 used where higher security margin (128-bit vs 110-bit) is required.
- **Trusted setup:** Maticz participates in multi-party computation (MPC) ceremonies (e.g., Hermez Perpetual Powers of Tau) so that setup toxicity is eliminated as long as at least one participant is honest. All ceremony transcripts are published on-chain.
- **Circuit language:** Circom 2 (constraint system), compiled to R1CS (Rank-1 Constraint System), proved via snarkjs (browser/Node) or RapidSNARK (C++, GPU-accelerated, ~10x faster proving).

zk-STARKs (Scalable Transparent Arguments of Knowledge)

- **Proof system:** FRI (Fast Reed-Solomon Interactive Oracle Proof) based; no trusted setup (relies only on collision-resistant hash functions — post-quantum secure by assumption).

- **Proof size:** ~45–200 KB (larger than SNARKs) but verification is faster for large computations. StarkEx and StarkNet use the Cairo VM with STARK proofs.

- **Hash function:** Rescue-Prime or Poseidon (SNARK/STARK-friendly hashes operating over prime fields, ~20x cheaper in-circuit than Keccak-256).

- **Field:** Prime field Fp where p = 2^251 + 17*2^192 + 1 (StarkWare's STARK-friendly prime).

Parameter	Specification / Standard
SNARK proof size	~192 bytes (Groth16) / ~400–800 bytes (PLONK)
STARK proof size	~40–200 KB
On-chain verification cost	Groth16: ~270k gas; PLONK: ~350k gas; STARK: ~5M gas (but amortized over many txns)
Proving time (Groth16, 1M constraints)	~2–5 s (GPU) / ~20–60 s (CPU)
Trusted setup required	SNARKs: Yes (universal or circuit-specific); STARKs: No
Post-quantum security	SNARKs: No (elliptic curve DLP assumption); STARKs: Yes (hash collision resistance)

6.3 Cross-Chain Interoperability

Maticz implements cross-chain message passing using two primary standards:

- **LayerZero v2:** Ultra-Light Node (ULN) model. Each chain runs an Endpoint contract. Messages are attested by a configurable set of Decentralized Verifier Networks (DVNs) and an optional Executor for automatic message delivery. Message packets are hashed using keccak256; the hash is verified on-destination using the source-chain block header supplied by the DVN.

- **Chainlink CCIP (Cross-Chain Interoperability Protocol):** Risk Management Network (RMN) provides a secondary validation layer that can halt transfers if anomalous activity is detected. Supports arbitrary message data + token transfers in a single atomic transaction. Uses OCR 2.0 (Off-Chain Reporting) for oracle aggregation.

- **Wormhole:** Guardian Network of 19 validators (threshold: 13/19 must sign). Publishes Verified Action Approvals (VAAs). Used for EVM ↔ Solana bridging where LayerZero has limited SVM support.

7. Smart Contract Engineering

7.1 EVM Contract Standards & Patterns

All Ethereum-compatible contracts are written in Solidity (≥ 0.8.20) with explicit pragma locks. We enforce the following engineering standards on every deployment:

Parameter	Specification / Standard
Language version	Solidity ^0.8.20 (or pinned minor version); Vyper 0.3.x for critical vault logic
Compiler optimization	Solc optimizer enabled; runs: 200 (deployment cost vs. call cost tradeoff); via-IR pipeline for complex contracts to enable Yul-level optimization
ABI encoding	ABI v2 (abi.encode / abi.encodePacked); structs and dynamic arrays handled via ABIEncoderV2 (default in 0.8.x)
Access control	OpenZeppelin AccessControl (role-based RBAC) or Ownable2Step (two-step ownership transfer to prevent address typos)
Upgrade pattern	EIP-1967 Transparent Proxy or UUPS (EIP-1822) for upgradeable contracts; Beacon Proxy for multi-instance upgrades
Reentrancy protection	ReentrancyGuard (mutex flag); checks-effects-interactions pattern enforced at code-review level
Integer safety	Solidity 0.8.x native overflow revert; SafeMath deprecated but safe casting via OpenZeppelin SafeCast for type conversions
Gas optimization	Packed storage slots (multiple variables in 1x 32-byte word); calldata vs memory distinction; unchecked blocks for loop counters where overflow is provably impossible

7.2 Token Standards — EVM

Maticz implements the full suite of Ethereum token standards with extensions:

- **ERC-20:** Fungible tokens. Extended with ERC-20Permit (EIP-2612) for gasless approvals via secp256k1 signatures (EIP-712 typed data). ERC-20Votes for on-chain governance weight snapshots using EIP-5805 delegation.

- **ERC-721:** Non-fungible tokens. Extended with ERC-721A (Azuki) for gas-efficient batch minting (single storage write for contiguous IDs). ERC-721Royalty (EIP-2981) for standardized royalty signaling (basis points on secondary sales).

- **ERC-1155:** Multi-token standard (fungible + non-fungible in a single contract). Supports batch transfers (safeBatchTransferFrom) and batch balance queries for gas efficiency in gaming and marketplace contexts.
- **ERC-4626:** Tokenized Vault Standard. Standardizes yield-bearing vault share accounting (convertToShares / convertToAssets). Used for all lending protocol share tokens and yield aggregators.

- **ERC-6551:** Token Bound Accounts. Each NFT can own an Ethereum account (ERC-4337 smart account), enabling NFTs to hold assets independently. Used in gaming for character inventory ownership.

7.3 Solana Program Architecture

Solana programs are compiled to BPF bytecode and deployed to the BPF loader (BPFLoaderUpgradeab1e11111111111111111111111). We use the Anchor framework (Rust) for safe account validation, discriminator-enforced deserialization, and IDL generation.

- **Account model:** All state is stored in accounts (not contract storage). Programs are stateless; they receive account lists as instruction inputs. Accounts have a fixed rent-exempt lamport balance proportional to their byte size (currently ~6.96 lamports/byte/epoch).

- **Token standard:** SPL Token (fungible) and SPL Token-2022. Token-2022 extensions used: TransferFee (on-chain royalty collection), ConfidentialTransfer (ElGamal-encrypted balances using twisted Edwards curve), InterestBearingMint (continuously compounding interest stored as a rate in basis points), and PermanentDelegate (immutable authority for compliance use cases).

- **Program Derived Addresses (PDAs):** Deterministic account addresses derived via SHA-256(seeds || program_id) with a bump seed to ensure the address is off the ed25519 curve. Used for all protocol-owned state (vaults, pool accounts, escrow accounts).

- **Cross-Program Invocation (CPI):** Programs invoke other programs via invoke() or invoke_signed() (for PDA-signed CPIs). CPI depth limit: 4. All CPI calls validate AccountInfo ownership to prevent spoofing.

7.4 Security Audit Standards

Every production deployment undergoes a minimum two-round audit process:

- **Static analysis:** Slither (Trail of Bits) for Solidity; Semgrep rules for custom pattern detection. Mythril for symbolic execution (bounded model checking). Findings are classified as Critical / High / Medium / Low / Informational per CVSS 3.1 scoring.

- **Manual audit:** Engagement with Tier-1 firms (Certik, Halborn, OtterSec for Solana). Audit scope includes economic attack vectors (flash loan manipulation, oracle sandwich attacks) in addition to code vulnerabilities.

- **Formal verification:** For core vault and AMM contracts, we apply formal verification using Certora Prover (CVL specification language) to prove invariants (e.g., total supply conservation, monotonic price curves) hold for all possible execution paths.

- **Invariant fuzzing:** Foundry's forge test with stateful fuzzing (forge-std/StdInvariant.sol). 10,000+ runs per invariant. Echidna (property-based fuzzer) for critical arithmetic properties.

8. DeFi Protocol Engineering

8.1 Lending & Borrowing Protocols

Our lending protocol architecture follows the Compound v3 (Comet) and Aave v3 designs with enhancements for capital efficiency and risk isolation.

Interest Rate Model

We implement a dual-slope interest rate model (kinked model): For utilization U ≤ Uoptimal: borrowRate = baseRate + (U / Uoptimal) * slope1. For U > Uoptimal: borrowRate = baseRate + slope1 + ((U - Uoptimal) / (1 - Uoptimal)) * slope2. The supply rate is: supplyRate = borrowRate * U * (1 - reserveFactor). Parameters (baseRate, slope1, slope2, Uoptimal, reserveFactor) are set per-asset by governance.

Collateral & Liquidation

- **Loan-to-Value (LTV):** Maximum borrow value as % of collateral value (e.g., ETH: 80%, USDC: 90%).
- **Liquidation Threshold (LT):** Threshold above which a position is eligible for liquidation (e.g., ETH: 82.5%).
- **Health Factor:** HF = Σ(collateral_i * LT_i) / totalDebt. If HF < 1.0, liquidation is triggered.
- **Liquidation mechanism:** Dutch auction-style close factor (max 50% per liquidation by default) to prevent toxic cascades. Liquidation bonus (e.g., 5%) incentivizes bots. Bad debt socialization via the reserve fund if collateral is insufficient.
- **Oracle:** Chainlink price feeds (8 decimal precision, heartbeat 1 h, deviation threshold 0.5%). Secondary Pyth Network feed used as circuit breaker; if feeds diverge > 5%, protocol pauses borrowing.

8.2 Staking & Yield Farming

Staking contracts implement a continuous rewards distribution model derived from Synthetix's StakingRewards:

rewardPerToken() = rewardPerTokenStored + (lastBlockTime - lastUpdateTime) * rewardRate * 1e18 / totalSupply. earned(account) = stakedBalance * (rewardPerToken() - userRewardPerTokenPaid) / 1e18 + rewards[account].

Rewards are updated on every state-changing call (stake, withdraw, getReward) via a modifier. This design avoids iterating over all stakers (O(1) per user regardless of pool size).

- **Liquidity mining:** Emissions follow a configurable decay schedule (linear, exponential, or epoch-based Gauges as in Curve veCRV model).

- **Vote-Escrow (ve) model:** Users lock governance tokens for up to 4 years; voting power = lockedAmount * (remainingLockTime / maxLockTime). Non-transferable veTokens direct emissions to gauges via weekly voting. This aligns long-term incentives with protocol health.

9. Infrastructure, DevOps & Security

9.1 Node Infrastructure

Parameter	Specification / Standard
Ethereum archive node	Erigon (flat-file state DB, ~2.5 TB vs Geth's ~12 TB for archive). eth_getStorageAt and eth_getLogs fully supported with sub-second response.
Ethereum full node (load balancer)	Nethermind (C#, high-throughput JSON-RPC). Multiple instances behind HAProxy with health-check failover.
Solana RPC node	Jito-Solana client. Bigtable backend for historical transaction lookup. WebSocket subscriptions for accountSubscribe and logsSubscribe.
Indexing	The Graph Protocol (subgraph with AssemblyScript mapping handlers). Goldsky Mirror for real-time CDC (Change Data Capture) to PostgreSQL.
Private mempool access	Flashbots MEV-Boost relay (Ethereum); Jito Block Engine bundle endpoint (Solana)

9.2 DevOps & Deployment Pipeline

- **CI/CD:** GitHub Actions workflows. On PR: Solidity compilation (Foundry forge build), test suite (forge test --gas-report), Slither static analysis, contract size check (< 24 KB EIP-170 limit). On merge to main: deploy to testnet (Sepolia / Solana Devnet) with automated integration tests.

- **Deployment scripts:** Foundry scripts (script/*.s.sol) with broadcast recording (broadcast/*/run-latest.json) for deterministic replay. CREATE2 factory (0x4e59b44847b379578588920cA78FbF26c0B4956C) for deterministic contract addresses across chains.

- **Infrastructure as Code:** Terraform for cloud resource provisioning (AWS EC2, VPC, Security Groups, KMS). Ansible for node configuration management. Kubernetes (EKS) for containerized services (indexers, bots, APIs).

- **Monitoring:** Prometheus + Grafana for metrics (node sync status, RPC response latency p50/p95/p99, transaction submission success rate, MEV bot PnL). PagerDuty alerting with runbook links for on-call engineers. Datadog APM for distributed tracing of multi-service agent workflows.

- **Secret management:** HashiCorp Vault (Transit for signing, KV v2 for API keys). AWS Secrets Manager for EKS pod secrets injection via External Secrets Operator.

9.3 Smart Contract Security Operations

- **Multisig governance:** All protocol admin functions (pause, parameter updates, upgrades) gated behind Gnosis Safe multisig. Minimum 3/5 or 4/7 threshold. Timelock contract (48–72 hour delay) between proposal and execution, giving users time to exit before parameter changes take effect.

- **Incident response:** Pre-defined runbooks for common attack vectors (oracle manipulation, reentrancy, price oracle flash loan attacks). Emergency pause key held in hardware wallet (Ledger Nano X) stored in geographically distributed custody. Post-incident disclosure follows the Rekt responsible disclosure standard (private report → 24h response → coordinated public disclosure).

- **Bug bounty:** Immunefi-hosted program with tiered rewards: Critical (economic loss > $1M potential) up to $500k USDC; High up to $50k; Medium up to $10k. Scope covers all production contracts and off-chain components with direct on-chain impact.

10. DAO Governance Framework

10.1 On-Chain Governance Architecture

We implement OpenZeppelin Governor (OZ Governor v1 and Bravo-compatible) with the following parameter template — all parameters are configurable per deployment:

Parameter	Specification / Standard
Voting delay	1 day (number of blocks from proposal creation to voting start; allows token accumulation before vote)
Voting period	5 days (number of blocks the vote is open)
Quorum	4% of total token supply (GovernorVotesQuorumFraction)
Proposal threshold	1,000 governance tokens (minimum tokens required to submit a proposal)
Timelock delay	48 hours (minimum time between successful vote and execution via TimelockController)
Vote counting	GovernorCountingSimple (For / Against / Abstain); Abstain counts toward quorum but not for/against
Vote weight	ERC-20Votes snapshot at voting start block (prevents double-voting via token transfer during vote)
Delegation	EIP-5805 (ERC-20Votes delegation; voting power is zero unless self-delegated or delegated to another address)

11. Compliance, Privacy & Regulatory Architecture

11.1 KYC / AML Integration

- **On-chain identity:** ERC-3643 Identity Registry stores Ethereum addresses linked to verified identity claims (ERC-735 ClaimHolder). Claims are issued by trusted Claim Issuers (KYC providers) and stored on-chain as signed attestations.
- **Off-chain providers:** Onfido, Jumio, or Sumsub for document verification + liveness detection. Results are published on-chain as signed claims (ECDSA signature over keccak256(address || claimType || expiry)). Claims expire and require periodic renewal.
- **OFAC screening:** Real-time address screening against OFAC SDN list via Chainalysis or TRM Labs API. Screening occurs on every transaction before signing; flagged addresses trigger an automatic halt and compliance team notification.
- **Travel Rule compliance:** TRISA (Travel Rule Information Sharing Architecture) or IVMS 101 standard for VASP-to-VASP beneficiary information sharing on transfers above $3,000 (FinCEN threshold) or €1,000 (FATF / MiCA threshold).

11.2 Privacy-Preserving Architecture

- **Confidential transfers:** SPL Token-2022 ConfidentialTransfer extension uses ElGamal encryption (twisted Edwards curve, Ristretto255) to encrypt token balances. The sender proves transfer validity via a ZK proof (σ-protocol for correct encryption) without revealing amounts.

- **Private smart contracts:** Aztec Network (ZK-ZK rollup) for fully private DeFi. Users interact via shielded accounts; all state transitions are proven via PLONK proofs. Suitable for institutional DeFi requiring trade privacy.

- **Data minimization:** Off-chain personal data is stored encrypted (AES-256-GCM) with user-controlled keys. On-chain data is limited to commitment hashes (keccak256 or Pedersen commitments). GDPR right-to-erasure is handled by deleting the encryption key (rendering data inaccessible).

12. Appendix — Key Standards Reference

Parameter	Specification / Standard
EIP-1559	Ethereum transaction fee market reform (base fee + priority fee)

EIP-4844	Proto-Danksharding: blob-carrying transactions for rollup DA
EIP-1967	Standard proxy storage slots (implementation, admin, beacon)
EIP-2612	Permit: gasless ERC-20 approvals via secp256k1 signatures
EIP-4626	Tokenized vault standard (yield-bearing share accounting)
ERC-3643	T-REX: regulated token standard with identity and transfer rules
ERC-6551	Token Bound Accounts: NFT-owned smart accounts
BIP-32/44/84	Hierarchical Deterministic wallet derivation paths
PLONK	Universal ZK proof system (no per-circuit trusted setup)
Groth16	Efficient ZK-SNARK (smallest proof, fastest verify; circuit-specific setup)
FRI / STARK	Transparent ZK proof (no trusted setup; post-quantum hash security)
LMSR	Logarithmic Market Scoring Rule (prediction market liquidity)
ERC-3156	Flash loan standard interface
IVMS 101	InterVASP Messaging Standard (Travel Rule data schema)
EIP-712	Typed structured data hashing and signing

END OF TECHNICAL SPECIFICATION

Maticz Technologies · 2026 · All technical parameters subject to revision