Vaulter Agents: Secure AI Delegation for NFT-Owned Smart Vaults

July 2025 <contact@vaultlayer.xyz>

Abstract

VaultLayer introduces **Vaulter Agents**, a secure and programmable delegation layer for **Smart Vaults** – decentralized vault accounts represented by NFTs. Each Smart Vault is a cross-chain wallet (Bitcoin + EVM) controlled by a threshold key, enabling users to hold assets and **delegate limited authority** to autonomous agents without surrendering private keys. We leverage Lit Protocol's threshold ECDSA network for non-custodial key management[1][2], ensuring that **no single node or party ever has the full private key** and signatures require a consensus of distributed nodes[3]. On-chain policy contracts and NFT ownership checks enforce strict controls on agent actions, making delegation **revocable and trust-minimized**.

Built on the ElizaOS framework[4], Vaulter Agents combine blockchain-aware AI with user-defined strategies to execute DeFi operations across chains. Supported strategies include **Bitcoin Liquid Staking** (staking BTC on CoreDAO for yield), **Smart DCA** (AI-driven dollar-cost averaging using Allora's price-prediction oracle), and **stablecoin yield farming** (e.g. on Aave or Morpho). The Smart Vault architecture enables **composable cross-chain strategies** – e.g. claiming BTC staking rewards on CoreDAO, swapping or lending assets on Ethereum, and rebalancing positions across networks – all orchestrated by an intelligent agent under cryptographic guardrails. We outline the system design, agent architecture, example strategies, security model, and token economics (the yield-bearing **vltCORE** token and forthcoming **VAULTER** governance token). This framework empowers technically literate investors, researchers, and DeFi developers to harness autonomous agents for cross-chain asset management with formal security and verifiability.

1. Introduction

Blockchain innovation has unlocked myriad financial opportunities across disparate networks, but managing assets and strategies across multiple chains remains complex and risky. Bitcoin, the most liquid crypto asset, has been largely siloed from the DeFi ecosystem due to bridging and custody challenges[5]. Staking BTC on Layer-1 platforms like CoreDAO, for example, yields rewards but *time-locks* the BTC and requires maintaining a certain BTC-to-CORE ratio to maximize yields[6]. Users face a dilemma: either **keep BTC in cold storage** (giving up on yield) or entrust it to centralized bridges and protocols (giving up security and control). Meanwhile, decentralized finance offers yield farming, lending, and algorithmic strategies on EVM chains, but coordinating these opportunities (and moving assets between chains) is cumbersome and often beyond the average user's capability[7].

VaultLayer's Smart Vaults address these issues by providing **NFT-owned vault accounts** that *span Bitcoin and EVM chains*[1]. A Smart Vault gives the user a unified cross-chain account – analogous to Ethereum's token-bound accounts (ERC-6551) which allow an NFT to control its own smart contract wallet[8] – but realized via an off-chain threshold key. Each Smart Vault NFT represents full ownership of a *decentralized key pair (PKP)* generated by the Lit Protocol network[1]. This PKP yields a Bitcoin address (for BTC custody) and corresponding addresses on supported EVM chains (CoreDAO, Ethereum, etc.), effectively making the NFT a **portable multi-chain wallet** under user control[9].

Crucially, VaultLayer introduces **autonomous agent delegation** on top of these vaults. Rather than expecting users to manually monitor and operate their vaults 24/7, **Vaulter Agents** can be assigned to perform tasks automatically – claiming rewards, reinvesting yields, executing trades – according to user-defined strategies. The core challenge is ensuring that such powerful agents **remain trust-minimized and under user control**. Traditional trading bots or yield optimizers often require users to hand over keys or funds, introducing custodial risk or reliance on off-chain logic without oversight. Vaulter Agents solve this by combining *on-chain enforceable policies* with *off-chain AI intelligence*: the user's NFT vault only executes actions that have been pre-approved via smart contract policies, and any agent transaction must be co-signed by Lit's threshold network which validates the NFT ownership and policy compliance at runtime[10]. In essence, the user can **delegate limited authority to an AI agent** while a combination of cryptography and smart contracts guarantees the agent stays within bounds.

This approach builds on prior research in account abstraction and cross-chain custody. By using threshold cryptography. By binding vault control to NFTs, we enable composability (vaults can be traded, tokenized or used as collateral) and an intuitive ownership model. By integrating an AI agent framework (ElizaOS[4]) we unlock intelligent automation: the agent can parse on-chain data, price oracles, and even natural language prompts to make decisions, all while **the user retains ultimate control** (the NFT owner can revoke or override the agent at any time).

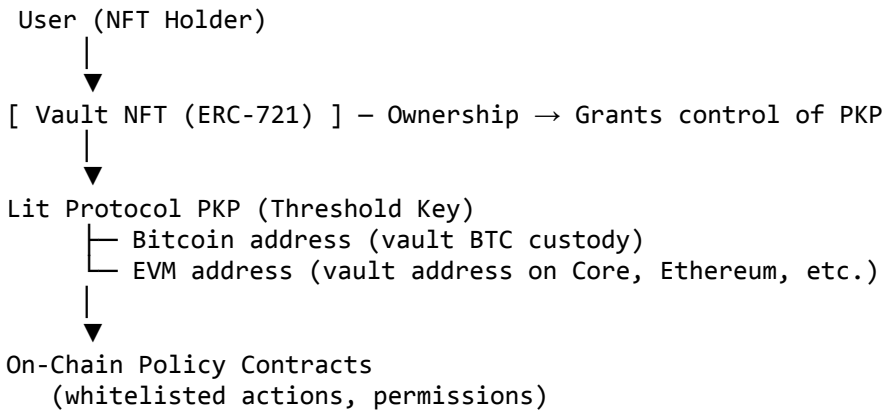
In the following sections, we detail the Smart Vault architecture, the Vaulter Agents design, example strategies (BTC staking, AI-driven DCA, stablecoin yield farming), the security model, various use cases, and the VaultLayer token model (including the **vltCORE** yield-bearing token and the upcoming **VAULTER** token). Our aim is to demonstrate a rigorous yet accessible design that marries the strengths of decentralized key management, smart contract enforcement, and

AI-driven automation to create **secure, user-owned DeFi agents**. This paper targets technically savvy investors, researchers, and developers. We employ clear explanations, architecture diagrams, and a focus on how each component contributes to a safer and smarter cross-chain DeFi experience.

2. Smart Vault Design

Smart Vaults are the foundational abstraction of VaultLayer – they are **programmable multi-chain vault accounts represented by ERC-721 NFTs**[1].

Below is a diagram summarizing the Smart Vault architecture:



Each Smart Vault comprises several tightly-integrated pieces:

- **NFT Ownership:** An NFT (non-fungible token) that represents ownership and control of the vault[14]. Possession of this NFT (in a user's wallet) is what gives the user the right to operate the vault. Transferring the NFT effectively transfers the vault and its contents – making vaults *tradeable and transferable* on secondary markets[15]. (This property has enabled a novel **NFT loan market** on CoreDAO where users borrow against vault NFTs[16], see Use Cases).

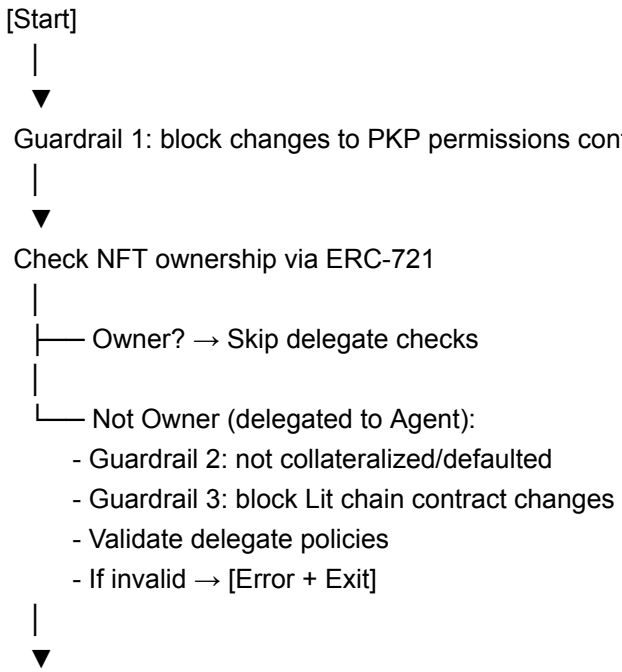
- **Decentralized Key (PKP):** A **Programmable Key Pair (PKP)** from Lit Protocol is minted and bound to each vault[9]. Lit Protocol uses distributed key generation (DKG) and threshold signing to create an ECDSA keypair that is **never fully assembled in one place**[12]. Instead, key shares are held by a network of independent nodes in secure enclaves (TEE), and any signing operation requires a threshold (e.g. 2/3 majority) of nodes to cooperate[3]. This yields security akin to a multi-sig with dozens of participants – no single node can misuse the key, and the private key remains secret even as signatures are produced. The **public key** of the PKP is used to derive the vault's blockchain addresses: e.g. a Bitcoin Taproot address and an Ethereum-like address for EVM chains[9]. Thus, one PKP gives the vault an identity on Bitcoin and across multiple Ethereum-compatible chains.

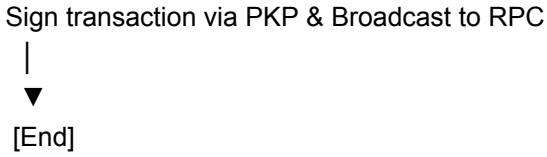
- **Vault Addresses (Bitcoin + EVM):** From the PKP's public key, VaultLayer derives a native SegWit BTC address and an address on each supported EVM chain (CoreDAO, Ethereum mainnet, Arbitrum, etc.)[9]. The vault can hold BTC directly on L1, as well as tokens on EVM chains, under the *same* cryptographic owner key. This design achieves **chain-abstraction**: the NFT owner effectively has a single vault that spans chains. For

example, a user's Vault NFT might correspond to BTC address `bc1...abcd` on Bitcoin and address `0x1234...abcd` on Ethereum (both derived from the same PKP public key). The Lit network will sign a Bitcoin transaction or an Ethereum transaction with the same decentralized key, under the hood, when authorized. *This is conceptually similar to ERC-6551 token-bound accounts* which give each NFT a wallet address[8], but here the "account" extends to Bitcoin and is secured by off-chain threshold cryptography rather than an on-chain contract.

- **Allowed Actions (Lit Actions):** Smart Vaults are *not* general-purpose externally owned accounts that can sign anything arbitrarily – instead, VaultLayer restricts them to a **whitelisted set of actions** implemented as **Lit Actions** (which are essentially secure Javascript snippets running in Lit's decentralized runtime)[17]. The current actions permitted for vaults include (with content identifiers registered on-chain):
    - **`signBitcoinTx`** – allows the vault to sign a Bitcoin transaction (e.g. to stake or unstake BTC in CoreDAO's contract, or redeem BTC).
    - **`callContract`** – allows the vault to call an arbitrary smart contract function on an EVM chain (used for things like transferring ERC-20 tokens, executing swaps on a DEX, claiming rewards, etc.).
    - **`coinTransfer`** – allows sending native chain cryptocurrency (e.g. CORE or ETH from the vault address).
    - **`decryptSecrets`** – allows the vault to decrypt stored secret data (used for fetching encrypted strategy parameters or off-chain data, e.g. an agent's prompt or a user's Telegram handle for notifications).
    - **`delegate`** – allows the vault owner to **delegate** certain permissions to an agent by approving specific Lit Action CIDs under on-chain policy constraints[18].

  The following diagram showcases the whole validation process performed on the *callContract* LitAction:





These Lit Actions are analogous to a restricted instruction set or syscalls for the vault. Each action's code is deployed to IPFS (content-addressed) and the hashes are registered in the VaultLayer contracts[19]. This means vault behavior is transparent and auditable: anyone can retrieve the code by its CID and review exactly what signing logic or contract call is allowed[20]. The **`delegate`** action is particularly important – it is how a Vaulter Agent is given permission to act on the vault's behalf in a controlled manner (more in the next section).

- **Policy Contracts & Permissions:** Every vault enforces an on-chain policy that ties the above components together. When a Lit Action is invoked by the vault's PKP, the Lit network performs checks before executing it:
- **Ownership check:** The Lit runtime runs `checkLitAuthAddressIsERC721Owner.ts`, a script that verifies the Ethereum address calling the action actually owns the Vault's NFT[21]. In other words, only the NFT holder (or an authorized delegate) can initiate vault actions. If the NFT is in Alice's wallet, Bob cannot trick the vault into signing anything because the Lit nodes will see Bob isn't the NFT owner.
- **Allowed action check:** The PKP (key) has an associated **PKPPermissions** contract on Lit's side. Each Lit Action CID must be explicitly enabled for that PKP by an owner transaction[22]. If an action or agent tries to execute a Lit Action that hasn't been whitelisted for this vault, the attempt fails. By default, only a minimal set (like `signBitcoinTx`, `callContract`, etc.) are enabled, and new ones (e.g. agent delegation actions) must be added by the NFT owner's consent.
- **No raw signing:** Notably, VaultLayer *disables any generic signing* (Lit's `signEcdsa` is turned off for these PKPs)[23]. This prevents a scenario where an agent could ask the vault to sign an arbitrary transaction payload (which might be a malicious transfer). All signatures must go through the vetted Lit Actions, which ensure context and policy are checked. This is an "anti-rug" safeguard: even if an agent's off-chain logic is compromised, it cannot directly obtain a valid signature to steal funds – it can only call the pre-built functions which have constraints.

Through these mechanisms, Smart Vaults achieve **non-custodial yet verifiable** operation. The user never has to trust a VaultLayer server or give up control: the private key is fragmented among independent nodes, and every transaction coming from the vault can be traced back to an authorized action. The vault's entire "agent policy" (which tools can it use, what contracts can it interact with, what limits are set) is encoded on-chain or via content-addressed code, providing full transparency[20][24]. If desired, the user (or anyone) can audit a vault's permissions using VaultLayer's scripts and the Lit Explorer[25] to confirm the vault cannot do anything beyond its intended design.

This design draws inspiration from Ethereum account abstraction (e.g. smart contract wallets that enforce rules) and threshold custody systems. By binding vault control to an NFT token, we get the benefits of **composability** – vaults can participate in other protocols. For instance, a Vault NFT can be listed on an NFT marketplace, tokenized or used as collateral in an NFT

lending platform (as already demonstrated on CoreDAO's lending market)[16]. By using **Lit's threshold DKG network**, we get a robust distributed security model: comparable to systems like Threshold Network's tBTC which used threshold ECDSA to manage BTC in DeFi[11]. In fact, VaultLayer worked closely with Lit to add Bitcoin support to their network[26], enabling signing of BTC transactions under the same security umbrella as EVM transactions.

In summary, a Smart Vault is a **decentralized, cross-chain wallet** owned by an NFT and governed by both code and cryptography. It allows users to stake and hold BTC with full custody (the BTC is in an address derived from their key, not a third-party)[27], and simultaneously interact with EVM DeFi (via the linked address) without needing separate accounts. Yet all interactions can be automated and safeguarded. The stage is now set to introduce the **Vaulter Agent** – the AI "brain" that can be given the keys (in a limited, safe way) to operate this vault on the user's behalf.

3. Vaulter Agents Architecture

Vaulter Agents are autonomous agents designed to manage Smart Vaults according to user-defined strategies, under strict security constraints. The architecture of Vaulter Agents combines off-chain AI processing (for intelligence and decision-making) with on-chain enforcement (for security and transparency). Below we break down how Vaulter Agents are built and how they operate:

- **ElizaOS Integration:** Vaulter Agents are built on **ElizaOS**, an open-source agent operating system for web3 applications[4]. ElizaOS provides a framework for creating AI agents (powered by large language models and other AI tools) that can read/write blockchain data and interact with smart contracts in a controlled manner. By leveraging Eliza, each Vaulter Agent benefits from an existing "cognitive core" – the agent can plan actions, maintain memory, and interact with web3 primitives through Eliza's libraries[28][29]. This means the agent can understand high-level instructions (e.g. "accumulate more BTC if the price dips 5%") and break them into on-chain transactions as needed. The key benefit of ElizaOS is that *every aspect of the agent is under user control and transparent*, since Eliza agents are essentially programs configured by the user[4]. The agent's code (or prompt) can be inspected, and Eliza facilitates safe plugin of web3 functionality.

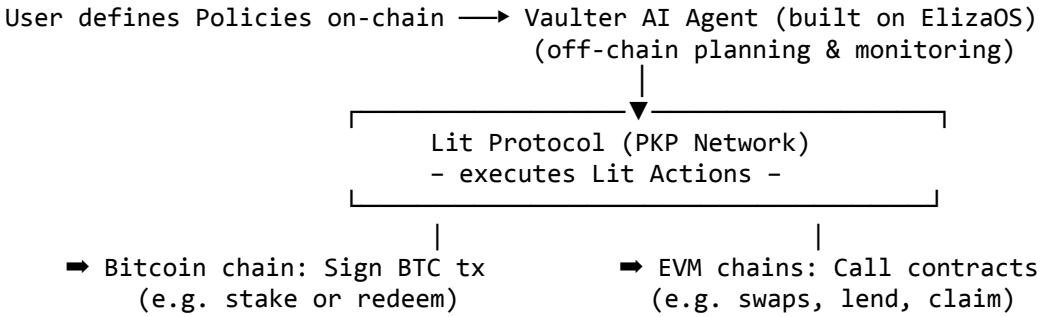
- **Delegation via PKP and Policies:** When a user decides to delegate their vault to a Vaulter Agent, they use the `delegate` Lit Action to grant the agent's specific Lit Action code permission on their vault's PKP[18]. In practice, VaultLayer provides preset agent "toolkit" actions (also content-addressed and audited) that the agent will invoke. For example, an agent might have a Lit Action CID for "execute DCA strategy" which internally calls `callContract` to buy BTC on an exchange. The user's NFT, through a VaultLayer on-chain transaction, adds that CID to the PKP's allowed list and sets parameters like which contract addresses and function signatures the agent is allowed to call (this is done in a **Tool Policy contract** for the agent)[30]. The agent itself does *not* receive the private key or any direct signing capability – it can only request the Lit network to execute certain actions, which will only succeed if the NFT owner approved them. This is a form of **principle of least authority**: the agent gets just enough permission to do its job, nothing more.

- **Agent Off-Chain Components:** The Vaulter Agent runs as a service off-chain (this could be a cloud function, a node server, or a decentralized execution node in the future) listening for triggers. It uses ElizaOS to maintain state and interact with the Lit network. For instance, the agent might subscribe to price feed updates or on-chain events. VaultLayer has also introduced a **Lit Event Listener** integration, meaning agents can react to on-chain events across multiple chains in real-time[31][32]. When conditions meet the user's strategy (e.g. a scheduled time for DCA, or rewards available to claim), the agent formulates the necessary transaction steps.

- **Execution Flow:** Once an agent decides to execute a strategy on the vault, the flow is as follows:

- **Trigger & Strategy Evaluation:** The agent detects a trigger (time to DCA, or yield threshold reached, etc.) and evaluates the strategy rules. It may use AI inference here – e.g. query Allora's price prediction oracle to decide if market conditions are favorable[33][34]. The strategy logic is either coded or provided as a prompt that the agent interprets.
- **Lit Action Invocation:** The agent then calls the appropriate Lit Action through the Lit SDK. For example, to claim staking rewards and reinvest, it might call a custom Lit Action that bundles a `callContract` to CoreDAO's staking contract (to claim CORE) followed by another `callContract` to swap CORE to BTC, etc. The agent does **not** sign a transaction with any private key; instead it sends a request to Lit nodes: "please execute Lit Action X using vault PKP Y".
- **On-Chain Policy Enforcement:** The Lit network receives this request. It checks: (a) Is the requester authenticated as the vault's delegate? (Lit can require the request to be signed by the NFT owner's session key or have other auth, depending on setup); (b) Is Action X allowed for PKP Y (per the PKPPermissions)?; (c) Does the action's code itself have any internal checks (e.g. limit values, allowed target addresses)? VaultLayer's Tool Policy contracts can, for instance, restrict that an agent can only interact with *specific protocols* or *spend up to a certain amount*. If all checks pass, the Lit nodes achieve consensus to execute the action.
- **Threshold Signing:** The Lit nodes run the Lit Action code in their TEEs. This code will typically assemble a transaction (BTC or EVM) for the vault and then call Lit's `sign` function to threshold-sign it[35]. Since the code is deterministic and agreed upon, the network produces a valid signature if quorum is reached. For an Ethereum transaction, it may also directly dispatch it to the chain (Lit can transmit the signed tx to the network).
- **Transaction Execution:** The transaction (e.g. a call to a DeFi protocol, or a BTC transfer) is executed on the respective blockchain from the vault's address. From the perspective of the blockchain, it's just the vault doing a normal operation; the difference is that it was initiated by the agent and authorized by the Lit network.
- **Feedback & Learning:** The agent receives confirmation of success or failure. If enabled, it can use this to update its state or notify the user. Vaulter Agents provide **Telegram updates** to users by default[28] – for example, "✅ Vault 123 claimed 0.5 CORE and swapped to 0.0008 BTC, new BTC balance X". Over time, the agent could even learn from performance (e.g. adjust DCA frequency if it notices better outcomes, subject to user approval).

The architecture can be visualized as follows:



Key Features of Vaulter Agents:

- **Secure by Design:** The agent operates strictly within the **on-chain permissions** of the vault[36]. If an agent tries to do something outside its scope, the transaction simply won't be signed. This mitigates risks from AI errors or "hallucinations". For example, an LLM might *suggest* swapping to an unsupported asset, but the on-chain policy would reject any call to an unapproved contract – the agent cannot go rogue. The agent also cannot withdraw assets to an arbitrary address unless that address is allowed by policy (generally only the owner or specific yield contracts are allowed). In effect, the vault's smart contracts sandbox the agent.

- **Automation & Intelligence:** Vaulter Agents can run 24/7 and execute **daily or even hourly strategies without user input[37][38]**. They incorporate AI tools for decision-making – for instance, using Allora Network's AI oracle, the agent gets **context-aware price forecasts** for BTC/ETH, etc., generated by an ensemble of models[39]. This allows the agent to do *predictive* moves (e.g. pausing DCA if a significant drop is predicted, or increasing allocation when an uptrend is likely). Allora's decentralized oracle provides short-term forecasts that can shift DeFi from reactive to proactive[40]. As noted in Allora's research, **retail traders gain access to state-of-the-art price prediction and risk assessment** which can enhance decision-making[13] – Vaulter Agents leverage this capability for users' benefit.

- **User-Friendly and Transparent:** Users interact with Vaulter Agents through simple configuration. One can **delegate to the agent via the app UI in a few clicks**[41], choosing a strategy template and setting parameters (e.g. "DCA $100 daily if price < $30k"). The agent provides human-readable updates on actions taken (via messaging integration), so the user is never in the dark about what the AI is doing[28]. Moreover, because the agent's moves are actual on-chain transactions from the vault, everything is traceable on block explorers. The combination of on-chain transparency and off-chain AI means the user can verify outcomes and hold the agent accountable to its code.

- **Modular and Extensible:** Thanks to ElizaOS, Vaulter Agents support **modular strategy plugins and custom prompts[29]**. As new DeFi opportunities or chains emerge, developers can add new tools (e.g. integration with a new DEX or lending protocol) as Lit Actions and update the agent's skillset. The architecture is not limited to BTC strategies – it is designed to evolve. For instance, one could plug in an **LLM planner** that converses with the user: "How would you like me to balance your portfolio this week?" and generate a plan, which is then executed under the safe policy framework. We

anticipate a library of agent "skills" (chain-specific actions, trading strategies, risk management routines) that advanced users or community developers can contribute, all gated by the same rigorous permission controls.

- **Single Agent, Multiple Vaults:** Traditional on-chain automation often requires deploying a separate agent per user or per strategy, which doesn't scale. Vaulter Agents are designed to be **multi-tenant and scalable** – a single agent service can manage many vaults, since it doesn't hold private keys and only triggers Lit network operations. Each vault's rules are distinct, but the agent's AI can handle context switching. This means VaultLayer can operate an "agent platform" that efficiently serves numerous users, and *costs are amortized*. From the user's perspective, they still have their personal agent, but under the hood it's an orchestrated system. This is in contrast to naive approaches where each agent might need its own key or contract (the "1-wallet-per-agent" anti-pattern that doesn't scale[42]).

Vaulter Agents effectively realize a vision of **self-driving vaults**: users set the destination (their financial goals and risk parameters), and the agent drives the vault accordingly, within guardrails. We solved the trust problem by introducing *delegated vaults with enforceable policies*[43] – thereby converting what would be a risky proposition ("let an AI trade my crypto") into a **verifiable, constrained, and user-revocable service**.

4. Strategies Executed by Vaulter Agents

We now describe three primary strategies that Vaulter Agents can execute for Smart Vault owners. These demonstrate the breadth of possibilities – from maximizing Bitcoin yields to intelligent investment automation and cross-chain yield farming. Each strategy highlights how the agent operates across different protocols and chains on behalf of the user.

a. Bitcoin Liquid Staking (BTC Staking on CoreDAO)

One flagship use-case of VaultLayer is enabling Bitcoin holders to earn yield via **CoreDAO's Satoshi Plus consensus** (which rewards BTC staking contributions) **without giving up custody**. In this strategy, a user stakes BTC from their Smart Vault into CoreDAO's L1 staking contract and in return earns CORE token rewards. Normally, staked BTC on CoreDAO is locked for months and users rarely hit the optimal 1:n BTC:CORE ratio for max rewards[6]. VaultLayer improves this by: 1) Pooling and **tokenizing the yield** via the **vltCORE** aggregator, and 2) Using an agent to automate reward claims and rebalancing.

Process: The user mints a Smart Vault and deposits (locks) some BTC into it for staking. Via a Bitcoin transaction (via a time-locked CLTV script[27]), the BTC is staked on CoreDAO's network. The VaultLayer protocol issues the user **vltCORE tokens** (ERC-20) which represent their share of the pooled staking position[44]. vltCORE implements the ERC-4626 vault standard[45], meaning it's a yield-bearing token that accrues value as rewards come in. Meanwhile, on the other side, CORE holders can deposit CORE into the protocol (receiving vltCORE as well) to help reach that 1:n ratio[44][46].

By using Vaulter Agents for BTC staking, users achieve **liquid staking** and yield optimization: they keep custody, earn CORE yields at the highest tier, and can get liquidity by borrowing against or selling their vault NFT or vltCORE tokens[51][16]. All the while, the tedious tasks (checking when to claim, initiating claims, re-staking CORE, monitoring ratio) are handled by the

agent autonomously. This strategy essentially transforms passive BTC into a productive asset in DeFi, *securely bridging Bitcoin to other chains*. As noted, VaultLayer leverages Lit's keys to create an *"offchain Bitcoin smart account"* that can interact with DeFi[2] – the BTC staking use-case is a prime example of this cross-chain capability.

b. Smart DCA (AI-Powered Dollar-Cost Averaging)

Dollar-cost averaging (DCA) is a popular investment strategy where one buys a fixed amount of an asset on a regular schedule, regardless of price, to mitigate volatility risk. VaultLayer's Vaulter Agents enable a **Smart DCA** strategy for accumulating Bitcoin (or other assets) using AI signals to enhance the timing and size of purchases.

Basic Idea: Instead of blindly buying say $100 of BTC every day, the agent uses an **AI price prediction oracle (Allora Network)** to decide *when* to execute the buy or *whether to adjust* the amount/frequency. For example, if the AI oracle strongly predicts a short-term price drop, the agent might delay today's purchase for a better price tomorrow; or if a rally is predicted, the agent might execute earlier to catch the upswing.

Allora Oracle: The Allora Network provides **decentralized AI-driven price forecasts** as an oracle service[33]. Dozens of machine learning models continuously retrain on market data and output a consensus forecast (e.g. "BTC price 24h from now likely +2%")[39]. This feed is available on-chain via an API. Research by Allora suggests that such AI-driven inference networks, by aggregating many models and using context-aware techniques, can outperform individual predictors and provide reliable signals[13][52].

This strategy showcases the **intelligent behavior** of Vaulter Agents. The agent isn't blindly executing a script; it's analyzing data (AI oracle outputs) and making a judgment call within allowed boundaries. This is only possible due to the integration of an AI layer. Traditional DeFi automation (like simple DCA contracts) cannot easily incorporate such off-chain intelligence. Here, the off-chain agent can digest complex signals but still *act on-chain securely* to realize them.

From a technical standpoint, Smart DCA involves cross-chain interactions as well: the agent might fetch a price feed from one network and execute a trade on another. Since the vault's key works on multiple chains, the agent could even arbitrage between DEXs on different chains if that were part of the strategy (though that's beyond simple DCA). It highlights how **cross-chain rebalancing** and movement can be handled seamlessly: the same agent can move assets from CoreDAO to Ethereum (perhaps bridging some stablecoins via a bridge contract) if that's where a better price or yield is, then move back – all under user-defined rules.

In summary, Smart DCA is "set it and forget it" Bitcoin accumulation enhanced by AI. It reduces the cognitive load on the user (no need to constantly watch markets) and ideally improves outcomes. The user can stop or modify it at any time by revoking the agent's delegation or changing the strategy parameters (all via their NFT control).

c. Stablecoin Yield Farming (Cross-Chain Yield Aggregation)

Beyond BTC-specific strategies, Smart Vaults with Vaulter Agents can serve as general **yield aggregators across DeFi protocols**. Consider a user who holds stablecoins (USDC, DAI, etc.) or other assets in their vault and wants to maximize interest with minimal effort. The agent can

be tasked with **stablecoin yield farming**, deploying capital to lending protocols or yield farms and auto-optimizing them.

Risk constraints: The user can impose that only certain protocols are allowed (e.g. Aave, Compound, Morpho – all well-audited). Perhaps exclude riskier farms. The agent can also be told to keep X% in a base safe platform and only move the rest. All these are configurable and enforced by policy (only addresses of approved protocols, etc.). - Over time, the agent accrues interest. The vault's stablecoin balance grows, or it mints yield-bearing tokens (like aToken, mToken) held in the vault. Because the vault is NFT-controlled, the user could even use the vault NFT representing that position as collateral elsewhere without withdrawing – an interesting composability angle (though care must be taken to not double-count collateral, outside scope here).

Automation of Yield Allocation: We refer to this as **yield allocation** – deciding where each dollar goes for optimal return. Vaulter Agents can be seen as user-specific yield optimizers, akin to a personal Yearn vault. The difference is the user retains custody and can override decisions. Also, the agent could incorporate custom risk assessments (perhaps an AI model rating the protocol safety or monitoring news). For example, if an exploit is reported for a platform, the agent could automatically withdraw funds (if the user had enabled such a safety trigger via a news oracle or a risk feed). This kind of responsive action is part of making the agent *intelligent and safety-conscious* beyond just yield numbers.

Cross-Chain Mechanics: The cross-chain nature is key. In the current DeFi landscape, yields differ widely by chain (Polygon's Aave vs Ethereum's Aave vs Optimism's, etc.). A user would need to manually bridge and move funds to chase these. VaultLayer simplifies that: the same vault can hold assets on all chains, and the agent can move them. Under the hood, when bridging, the agent might call a bridge contract (like Hop or Stargate) to move USDC, or use a DEX on one chain to swap to a bridgeable asset then swap back. Each step is again governed by what contracts are allowed. The user doesn't have to do anything – they just see in their dashboard that now some assets moved chain and yield is higher. This is **cross-chain rebalancing** in action, one of the highlighted capabilities of Vaulter Agents.

Overall, stablecoin yield farming via Vaulter demonstrates the platform's chain-agnostic design: the agent perceives a unified field of opportunities across chains and uses the vault's multi-chain reach to take advantage. It's worth noting the agent's actions (moving money around) do incur gas and possibly bridge fees – the agent will factor those in (e.g. not hopping chains for a tiny APY increase that doesn't cover fees). The Eliza AI layer could be involved in these decisions, potentially even using reinforcement learning to maximize net yield minus costs.

To conclude this section, Vaulter Agents are not limited to these strategies; they can also perform **swaps and arbitrage**, manage **LP positions in AMMs** (adding/removing liquidity when fees are optimal), or handle **governance actions** (e.g. vote with tokens in the vault). The three strategies above – BTC staking, AI-enhanced DCA, and yield farming – illustrate how the system handles *long-term investing*, *algorithmic trading*, and *lending yields* respectively. In all cases, the common thread is **automation with assurance**: the heavy lifting is done by an autonomous agent, but the user's assets are safeguarded by the vault's cryptography and the agent's leash (policies).

5. Security Model

Security is paramount in VaultLayer's design. By introducing AI agents into asset management, we must ensure that users do not inherit new risks. Below we analyze the security model, covering custody guarantees, agent restrictions, potential attack surfaces, and how they are mitigated.

1. Non-Custodial, Distributed Key Custody: At the core, user funds (BTC, ETH, etc.) reside in addresses controlled by a **threshold key** that no single party possesses. The Lit Protocol network operates under a *BFT-like security model* – it requires an honest majority of nodes (e.g. 2/3) to act to produce a signature[3]. This prevents rogue operators from unilaterally moving funds. Lit's DKG means even if an attacker compromised some nodes, they still could not reconstruct the private key[12]. Additionally, Lit rotates key shares periodically[54], adding resilience. This is similar to the security assumptions in decentralized custody solutions like tBTC (honest majority)[11]. Users therefore are not trusting VaultLayer or Lit *as a single custodian*; they trust a robust network with economic incentives and technical safeguards (TEE, stake slashing, etc. as described in Lit's whitepaper[3]). The **private key never leaves the enclave of Lit nodes** and never materializes fully, eliminating many traditional key theft vectors.

2. NFT Ownership as Root Authority: The linkage of vault control to an NFT provides a clear, on-chain source of truth for authorization. Only the holder of the NFT can approve agent permissions or directly invoke actions. This means if a user's wallet is secure, their vault is secure. Conversely, if the NFT is stolen (e.g. phishing), an attacker could gain control *akin to stealing a private key*. To mitigate this, standard wallet security best practices apply (hardware wallets, etc.). VaultLayer could also integrate **social recovery or multi-sig for NFT ownership** (e.g. requiring two keys to move the NFT) for additional safety, though that's an extension. On a positive note, NFT-based control means revocation is easy: the user can transfer the NFT to an isolated wallet to "freeze" agent access, or call a revoke function that removes all agent permissions from the PKP (VaultLayer provides a one-click "Emergency Stop" that unassigns any agent and halts automated actions). Because the Lit check always verifies NFT ownership on each action[22], the moment the NFT is not owned by the agent-authorized address, the agent loses power. This is a strong safety feature unique to tying control to an ERC-721.

3. Enforceable Policies & Least Privilege: Agents are given only the minimum scope they need. This is enforced by multiple layers:

- **Tool Contracts & Allowlists:** VaultLayer's Tool Policy contracts enumerate exactly which contract addresses and functions an agent can call, and even value or frequency limits. For example, the Smart DCA agent might be allowed to swap on Uniswap but only up to $X per day. If it tries to exceed, the on-chain policy will reject the transaction (the Lit Action will fail the check).

- **Content Restrictions:** Because Lit Actions code is fixed and auditable, one can ensure it *only performs intended operations*. The `delegate` Lit Action, for instance, might only allow granting permission to known agent CIDs (not an arbitrary malicious script). The agent's own Lit Actions (like "execute strategy Y") are coded to interact with specific protocols. This all reduces the surface for an agent to do something unintended. -

No Direct Key Access: The agent never handles private keys or raw signing ability. Even if the AI is subverted or behaves maliciously, it cannot sign a transaction except through the Lit interface, which imposes the above checks. This is critical: even a

compromised server running the agent cannot steal funds directly; it would have to trick the Lit network into signing something outside policy, which by design it cannot. The worst it could do is waste gas on failed attempts or, if it had some allowed destructive action like swapping all assets to a certain coin, it could do that *only if the user had mistakenly allowed it*. That's why VaultLayer's default policies are conservative – e.g., an agent cannot send funds to an arbitrary EOA, period. It could only ever deposit into known protocols or back to the owner.

4. Verifiability and Transparency: Security is bolstered by the fact that everything the agent is allowed to do is transparent to the user and community. The policy and code are public[55]. This enables **open security auditing**. The community can inspect the agent code for bugs or risks. Smart contract audits are part of VaultLayer's pipeline (both VaultLayer's own contracts and any Lit Actions). For instance, if an agent's Lit Action had a flaw where it might over-withdraw, that could be detected by reviewing the code on IPFS. VaultLayer has undergone security audits (and lists them in documentation[56]). Additionally, the underlying Lit Protocol has been through audits and a long beta period securing keys (with bounties to incentivize finding vulnerabilities). All these measures aim to ensure the entire stack – from cryptography to agent logic – is vetted.

5. Handling AI Specific Risks: AI brings unique challenges: hallucinations, unpredictable outputs, etc.[57]. VaultLayer's approach is to keep the AI on a *short leash*. The agent's freedom is constrained to choosing *when and which allowed action to execute*, not to invent new actions. If an LLM "thinks" the best strategy is to send funds to address X (not in policy), it simply cannot do it – the attempt will be blocked. If it hallucinates a wrong contract address, that address won't be in the allowlist, again blocked. So the worst an agent's AI mistakes can do is cause no-ops or suboptimal but still safe actions. We also utilize AI for monitoring the AI: for example, one could have a secondary watcher agent (with no signing power) analyze the primary agent's decisions for sanity and alert the user if something seems off (this is an area of ongoing R&D, not in initial release, but conceptually feasible with ElizaOS multi-agent setups). Furthermore, by logging all agent decisions and outcomes (and sending summaries to the user), the user can detect if the agent is underperforming or doing something unexpected and then intervene (pause or adjust it). This human-in-the-loop ability is important – it's not a black box hedge fund where the user has no clue what's happening; they can always see transactions and have the final say to stop automation.

6. Economic Security and Incentives: The Lit nodes have economic incentives to be honest (they can be slashed or not rewarded if they misbehave, and they stand to gain by the network's use). While specifics of VAULTER's role are not yet defined, VaultLayer will likely layer on its own token incentives (VAULTER) for agent operations (possibly staking by agents or rewards for certain security behaviors).

7. Comparisons and External Risks: We compare the security to alternatives:

- **Centralized Bots:** If a user gave their private key to a cloud trading bot, they face total loss risk if the bot is compromised. Vaulter Agent's design avoids that single point failure – the key stays sharded among >N nodes and cannot be exported.

- **Multi-sig with a bot signer:** If one tries to DIY by making a multi-sig and letting a bot be one signer, that bot could still collude to sign maliciously if quorum is small, or you add delay and friction if quorum is larger. Our threshold scheme effectively is an M-of-N

with M > N/2 across many nodes, which is far more secure while being transparent to the user (no manual signing needed).

- **Smart contract vaults (like Argent or Gnosis Safe) with modules:** Those allow adding modules with logic, but if a module (analogous to our agent) has a flaw, funds might be drained. In VaultLayer, even if the agent logic had a flaw, the threshold signing and off-chain nature means an attacker can't exploit that flaw to directly drain funds unless it can trick >2/3 of nodes. This is a subtle benefit: an on-chain wallet contract bug is immediately exploitable; an off-chain agent bug would likely manifest as an attempted unauthorized action that Lit nodes simply don't sign for because it breaks policy.

- **Bridge Risk:** Because VaultLayer does cross-chain, it does rely on bridging if moving assets between chains. We mitigate this by limiting to reputable bridges and possibly using native chain bridges (like using the vault's Bitcoin capability rather than wrapping BTC, or using canonical bridges where possible). Nonetheless, any cross-chain transfer of value carries risk of the bridge. This is a general cross-chain risk and is not unique to our agents, but we mention it. The agent can be programmed to avoid risky bridges and prefer atomic swaps (if available) or just maintain separate balances per chain to minimize bridging.

In conclusion, VaultLayer's security model is layered defense:

- **Crypto security** (threshold keys, TEEs, consensus),

- **On-chain security** (smart contract checks, open auditability),

- **AI safety** (policy constraints, monitoring, user oversight),

- **Economic safety** (incentives, potential insurance).

This comprehensive approach ensures that users can confidently delegate to Vaulter Agents. The design acknowledges the novel risks of autonomous agents and addresses them by combining proven techniques from distributed systems and smart contract security. By formalizing the agent's capabilities and verifications (effectively treating the agent as an unpredictable actor that must be tightly sandboxed), we significantly reduce the attack surface compared to naive automation.

The result is a system where a user could sleep with an AI agent managing their funds *and not wake up to a nightmare*. Instead, they wake up to a report of earnings and know that even while they slept, cryptographic guardians kept watch.

6. Use Cases

Vaulter Agents and Smart Vaults unlock a range of use cases across the DeFi and crypto landscape. We highlight a few, demonstrating the versatility of the platform for different user personas and scenarios:

- **"Hands-Off" Bitcoin Investor:** *Profile:* Alice holds BTC as a long-term investment. She wants to earn yield on it but doesn't trust centralized services. *Use Case:* Alice mints a Smart Vault NFT, stakes her BTC on CoreDAO via the vault, and delegates to a Vaulter Agent with the **BTC Liquid Staking strategy**. Now Alice is earning CORE rewards on her BTC and the agent automatically compounds her rewards into more BTC. She

receives weekly Telegram updates of how much her BTC position grew. If Alice needs liquidity, she has options: she could use her Vault NFT as collateral on a lending platform to borrow stablecoins (since the NFT represents staked BTC with yield)[16], or she could even sell the NFT to someone else (transferring the vault) instead of unstaking (which might have a time lock). The agent ensures her stake is always in the optimal tier and that her rewards are claimed and not sitting idle. Essentially, Alice achieves a "**self-driving BTC savings account**" that preserves the self-custody ethos of Bitcoin.

- **DeFi Power User (Yield Maximizer):** *Profile:* Bob is active in DeFi and holds assets across many chains. He wants the best yields but managing funds on 5 different chains is too time-consuming. *Use Case:* Bob deposits a mix of assets into a Smart Vault (e.g. some BTC, some ETH, some stablecoins). He configures a **Yield Farming agent** to manage stables and ETH for yield, and a **DCA agent** to accumulate more BTC. The yield agent deposits his USDC into Morpho on Ethereum and his ETH into Aave on Arbitrum, since those yield highest. When Optimism introduces a liquidity mining program with very high APY for ETH, the agent moves some ETH there to capitalize, then moves it back when it ends. The DCA agent uses the yield earned to periodically buy BTC on dips. Bob basically has a **personal yield manager** that scours opportunities and moves his capital, which he can track through a unified dashboard. Unlike using separate yield platforms (Yearn, etc.), Bob's Vault agent can do cross-platform moves that others might miss, giving him an edge. Meanwhile, Bob retains full control and can even manually intervene via the vault if he desires (e.g. instruct the agent via a prompt to avoid a certain protocol if he's skeptical about it).

- **DAO/Treasury Management:** *Profile:* A DAO (decentralized autonomous organization) has a treasury (in various assets) that it wants to deploy for returns without constant micro-management by members. *Use Case:* The DAO creates a Smart Vault controlled by a multisig (the NFT is held by the multisig or a time-locked contract controlled by DAO governance). They configure a Vaulter Agent to perform a set of strategies: provide liquidity in stablecoin pools, stake any idle ETH in liquid staking, etc. The agent can even be given a rule to send a portion of yield back to the DAO's operational wallet monthly. Because the vault's policy is transparent, DAO members can verify the agent cannot do anything weird like send funds to individuals. It's all encoded that it can only interact with whitelisted protocols. This setup gives the DAO a quasi-"employee" that works 24/7 optimizing the treasury, but with no salary and no risk of human corruption. The **autonomous treasury agent** could be a trend – DAOs employing AI agents for routine asset management under strict governance-approved rules.

- **Cross-Chain Arbitrageur:** *Profile:* Carol is an arbitrage trader needing to move quickly across exchanges on different chains. *Use Case:* Carol uses a Smart Vault with an agent that monitors price feeds and performs **cross-chain arbitrage**. Suppose a stablecoin is cheaper on BSC than on Ethereum; the agent can buy on BSC and sell on Ethereum near-simultaneously, using the vault's addresses on both chains. The threshold key ensures the trades execute (almost) atomically in the sense that one agent coordinates them. Profits accumulate in the vault. Here, the speed and coordination of an AI agent shine – it can react faster than Carol manually could. And since the vault is one unified entity on all chains, Carol doesn't juggle multiple wallets or worry about transferring funds in the middle of an arb (which could kill the opportunity). The agent effectively acts like a cross-chain market maker. This use case pushes the boundaries of Vaulter Agents

(it requires very fast decision-making and possibly higher frequency operation than daily tasks), but it's feasible especially with Lit's event listening for cross-chain events[32]. It also highlights how agents can bring liquidity and efficiency to markets (benefiting the ecosystem by tightening spreads).

- **Retail User – "Set and Forget" Crypto Portfolio:** *Profile:* Dave is a newcomer to crypto who wants exposure to BTC and ETH and a bit of DeFi yield, but doesn't want to actively trade. *Use Case:* Dave buys a Smart Vault NFT (perhaps some wallets will abstract this and he won't even realize it's an NFT under the hood). He deposits $10k: $5k becomes BTC in the vault, $5k into stablecoins. He then chooses a pre-built strategy that says: "Use stablecoins to earn yield until a dip of >10% in BTC or ETH occurs, then buy more BTC/ETH." The Vaulter Agent then handles it: it puts stables into Aave earning interest, watches the market. Two months later a dip happens, the agent withdraws some stables (losing a bit of interest but that's fine) and buys ETH at a 15% lower price. Dave didn't have to time the market; his agent did it calmly. Over a year, Dave ends up with more BTC and ETH than he started (thanks to buying on dips and yield in between). He experiences this through periodic summaries and maybe a web UI graph. Importantly, Dave never had to move funds across chains or interact with complicated DeFi UIs; the agent abstracted all that. This shows how Vaulter Agents can be packaged into **user-friendly robo-advisors** for crypto – with the differentiator that they are non-custodial and fully transparent, addressing a big trust gap in the "robo-advisor" concept.

- **Composable Vault NFTs:** Because Vault NFTs are standard ERC-721, an interesting use case is emerging NFT financialization. For example, suppose Erin is a sophisticated user who provided liquidity in a Vault. She now holds a Vault NFT which has certain assets. If she wants to exit the position, she could list the NFT for sale on a marketplace. The buyer of that NFT effectively buys the entire vault (with its assets and agent strategy continuing). This could create a secondary market for yield-bearing vaults, similar to selling a portfolio. Additionally, the NFT could be used in **NFTfi** protocols: Erin could borrow ETH by collateralizing her Vault NFT (which has clear value from its assets)[16]. If she fails to repay, the lender gets the NFT, thus the vault's contents. The agent even keeps managing during this period, which benefits whoever ends up with it. This use case wasn't possible with traditional static NFTs but becomes logical with Smart Vaults – each vault is like a productive business that can be transferred.

These use cases demonstrate the flexibility of VaultLayer's system to adapt to many scenarios: from passive investing to active trading, individual users to organizations, and even new financial primitives. Underpinning all of them is a common value proposition:

- **Autonomy:** The agent handles the day-to-day or minute-to-minute actions.

- **Security:** Users do not have to trust the agent beyond the code it's given; they hold the ultimate keys (via NFT and Lit's guarantees).

- **Composability:** Vault accounts can plug into the broader DeFi/Lego ecosystem (NFT marketplaces, Agent Tokens, lending, governance, etc., as any account could).

- **Cross-Chain Fluidity:** Value can seamlessly flow to where it's most effective, without the user manually bridging or switching wallets.

In essence, VaultLayer's Smart Vaults and Vaulter Agents together form a **secure automation layer for Web3**. Much like how self-driving cars aim to chauffeur people safely to destinations of their choice, Vaulter Agents aim to drive user assets toward financial goals safely and efficiently. The use cases above illustrate the broad highway of possibilities that opens up when you can trust an AI co-pilot with your crypto – with VaultLayer providing the guardrails to keep that journey safe.

7. Token Model

VaultLayer's ecosystem includes a dual token model: **vltCORE** (VaultLayer Core token) and **VAULTER** (VaultLayer's governance token). These tokens serve different purposes – one is a utility/liquidity token tied to the BTC staking strategy, and the other is envisioned as a governance and incentive token for the broader platform. We describe each in turn:

vltCORE (Bitcoin-CORE Staking Shares)

vltCORE is an ERC-20 token that represents a share of the VaultLayer BTC staking pool on CoreDAO[44]. As discussed in the BTC Liquid Staking strategy, when users stake BTC through VaultLayer, they effectively enter a pool where CORE tokens are also staked to maximize rewards. vltCORE is issued to users to denote their claim on this pooled position[44]. It is akin to a "receipt" or liquid staking derivative: - When you stake 1 BTC in a Smart Vault and it gets aggregated, you receive an equivalent value of vltCORE tokens. Similarly, if you deposit CORE into the pool, you get vltCORE. - The vltCORE token's value increases over time as rewards accrue. It's implemented as an **ERC-4626 vault token**[45], meaning the contract defines conversion between underlying assets and the share token. As CORE rewards are collected, the total underlying per vltCORE increases. Users can redeem vltCORE for the underlying (BTC and/or CORE in the pool) at any time subject to any staking lock periods. - vltCORE is **transferable and composable**[58]. Users can trade it or use it as collateral. For example, someone could swap vltCORE for USDT on a DEX if there's a market, effectively letting them exit their position without waiting for the BTC unlock. This provides liquidity to what would otherwise be a locked BTC stake. - The design of vltCORE includes automated rebalancing and reward distribution logic: it keeps track of the BTC:CORE ratio and splits rewards between BTC stakers and CORE depositors fairly[49]. If, say, CORE depositors are contributing more relative to target, they get a higher portion of rewards (which is reflected in how the share values adjust).

Utility of vltCORE: In the VaultLayer system, vltCORE serves as the **yield-bearing token** that users interact with if they want a simpler interface. Some users may never touch a Vault NFT directly but might just buy vltCORE tokens on an exchange to get exposure to BTC staking yield. It abstracts away the vault complexity for those who prefer a token model. Meanwhile, Vaulter Agents can also interact with vltCORE – for instance, an agent could decide to convert rewards to vltCORE and hold that, or to arbitrage vltCORE price if it drifts from its intrinsic value.

In summary, vltCORE aligns incentives of BTC stakers and CORE providers by giving them a common unit. It is the backbone of the **yield aggregation model** for CoreDAO staking[44]. Technically, it's similar to other liquid staking tokens (like stETH for Ethereum's staking) but unique in that it represents a multi-asset pool (BTC and CORE) and is managed by cross-chain logic.

VAULTER (Governance Token)

VAULTER is VaultLayer's governance token (planned, not yet released at the time of writing). While full details are pending final tokenomics announcements, we outline some potential roles of VAULTER based on typical DeFi governance tokens:

- **Governance:** VAULTER will likely be a governance token, allowing holders to vote on protocol parameters, new strategy approvals, fee rates, etc. For example, VAULTER holders might vote on adding support for new chains or assets for agents, or adjusting the target BTC:CORE ratio if CoreDAO changed its policy. Decentralized governance is important given VaultLayer's cross-chain scope; decisions like which bridges are considered secure enough for agents could be community-governed.

- **Fee Capture and Staking:** The platform may charge certain fees – e.g. a small performance fee on yields or a fee for using the agent service. These could accrue value to VAULTER. For instance, fees might go into a treasury or be used to buy back and burn VAULTER (common mechanism). Or VAULTER holders might receive dividends in the form of a portion of yields. Potentially, VaultLayer could implement an analog where VAULTER captures the "AI automation fee" value.

- **Incentives:** To bootstrap the ecosystem, VaultLayer might use VAULTER rewards. For example, early users who stake BTC or provide CORE might get VAULTER emissions (liquidity mining style) to incentivize participation (on top of natural yield). Also, running a Vaulter Agent (if decentralized in future) might require staking VAULTER to align interests, or agents could earn VAULTER for good performance. The presence of **LASER points and a future TGE (Token Generation Event) airdrop**[60] indicates that early adopters are being rewarded, likely with VAULTER when it launches.

- **Access or Discounts:** VAULTER could potentially be used to pay for agent services or get discounts. For example, if the platform introduces a subscription for advanced agent features, paying in VAULTER or holding VAULTER might reduce costs. Or certain advanced strategies might only be accessible to users staking a certain amount of VAULTER (this would align token value with platform usage).

- **Security Staking:** In the long run, VaultLayer might allow community members to run parts of the infrastructure and require staking VAULTER as collateral for honest behavior.

Given VaultLayer's emphasis on autonomy and AI, VAULTER could also find novel use cases: perhaps **governing AI parameters** (like voting to deploy a new AI model for agents if it's proven better), or funding agent R&D via the DAO, etc. The token essentially ties the community to the success of the "vault+agent" ecosystem.

Currently, without specific tokenomics announced, VaultLayer encourages community building via **LASER points** (an early user points program) to potentially distribute a portion of VAULTER fairly to contributors and users[60]..

In summary, **vltCORE** is the utility token that directly interfaces with one of VaultLayer's core services (BTC staking yields), whereas **VAULTER** is the broader platform token for governance, incentives, and capturing the value created by all services (including things like agent usage, cross-chain coordination, etc.). Together, they complement each other: vltCORE solves an

immediate functional need (liquidity and composability for staked BTC assets)[61], and VAULTER aligns long-term interests of the ecosystem's participants and those securing/using the network.

As VaultLayer expands to more strategies (perhaps a vault token for stablecoin pools, etc.), VAULTER's role as a meta-governance token will become even more important – steering a constellation of vault tokens and agent services.

The token model thus balances *pragmatism* (use a derivative token like vltCORE where needed for immediate utility) with *decentralization and incentive alignment* (use a governance token VAULTER to drive the project's evolution and reward contributors).

Users interested purely in yield might hold vltCORE, while users who believe in the growth of AI-managed vaults might hold VAULTER. Some will hold both, e.g., stake BTC and get vltCORE, then stake some of their VAULTER to participate in governance decisions about adding new yield strategies for the agent to deploy their BTC into – closing a virtuous loop where the community of token holders guides the system that in turn benefits the token economics.

8. Conclusion

VaultLayer's Vaulter Agents and Smart Vaults together present a novel paradigm for secure, autonomous asset management in the crypto space. We have detailed an architecture where **NFT-bound vault accounts** give users cross-chain control over their assets, and **AI-powered agents** can act on those assets within a rigorous security framework. This achieves a long-sought goal in DeFi: empowering users to reap the benefits of sophisticated, cross-chain strategies and 24/7 market monitoring *without* surrendering custody or relying on opaque intermediaries.

The key innovations can be summarized:

- **NFT-Owned Smart Vaults:** By representing vaults as NFTs, we inherit all the composability of NFTs (tradeability, use in other protocols) while binding it to real asset custody via threshold cryptography. This marries the concept of token-bound accounts (ERC-6551) with a multi-chain twist, effectively making NFTs into **full-fledged decentralized accounts**[8][9]. Users maintain self-sovereignty – your keys (sharded across nodes) and your coin.

- **Threshold Key Infrastructure:** Using Lit Protocol's threshold ECDSA network, VaultLayer ensures no single point of failure in key management[12]. This distributed trust model is **crucial for bringing Bitcoin into DeFi** securely[5]. It also means the vault's identity is consistent across chains, enabling seamless cross-chain actions without trusted bridges.

- **Programmable Delegation with Policies:** Vaulter Agents introduce a secure middle-ground between manual control and full automation. Unlike previous attempts at trading bots or asset managers, Vaulter Agents operate under **cryptographic and code-enforced constraints**. The on-chain policy layer is a powerful concept: it's like having a built-in compliance and safety officer that checks every move the agent makes. This gives users and developers confidence to explore AI-driven strategies, as the worst-case outcomes are limited by design.

- **AI-First Automation:** By incorporating ElizaOS and AI oracles (Allora), VaultLayer is **at the forefront of the AI+Web3 convergence**. The system shows how an LLM-driven agent can directly interface with decentralized finance – planning actions and executing them trustlessly. This architecture is extendable; today it's BTC and DeFi, tomorrow such agents could manage NFT portfolios, execute DAO treasury policies, or interact with real-world asset protocols, all with minimal tweaks. We foresee Vaulter Agents as part of a broader trend of **Autonomous Economic Agents (AEAs)** that will populate crypto ecosystems, making markets more efficient and lowering the barrier to entry for users to benefit from complex strategies.

- **Security & Formality:** Emphasizing security was not just a feature but a requirement for user acceptance. We demonstrated that every layer (key management, contract logic, agent behavior) has been meticulously secured or constrained, and that this model stands robust against many threats. Importantly, everything is **verifiable and auditable**, aligning with the scientific ethos of transparency and peer review. Users, or independent auditors, can validate the system's claims (e.g., inspect that no Lit Action can transfer funds to arbitrary addresses, or that the threshold signing indeed needs M-of-N nodes). In practice, this means the VaultLayer system can earn the kind of trust that DeFi protocols like Compound or Uniswap enjoy – trust in code, not in individuals.

- **Cross-Chain Composability:** VaultLayer doesn't view blockchains as silos; it treats them as a unified playground where assets can be moved and utilized where most effective. Smart Vaults abstract away the complexity of chain differences. This could spur more fluid capital flows, where, for example, Bitcoin liquidity can be dynamically allocated to wherever in DeFi it's most needed at the moment (via the agents). It aligns with the industry's vision of a multichain future with interoperability, but whereas many solutions rely on message passing or bridging alone, VaultLayer leverages a shared cryptographic key approach – offering a complementary path to cross-chain integration that can work alongside protocols like LayerZero or IBC.

In conclusion, **VaultLayer's Vaulter Agents represent a significant step toward autonomous finance** – a world where users can delegate tasks to AI agents with the same confidence and security as if they were doing them manually on a hardware wallet. This has profound implications: it can democratize access to advanced financial strategies (the AI can manage $100 with the same finesse as $1M, which is great for retail users), and it can free humans from constant monitoring while arguably improving performance by reacting faster and basing decisions on data-driven models.

For seasoned investors, this means opportunities for better yields and diversified strategies with less effort. For protocol researchers and developers, VaultLayer offers a case study in blending AI and crypto in a safe way – inviting further innovation in this direction (we anticipate research on agent coordination, on-chain verification of agent behavior, etc., building on this foundation). For the DeFi ecosystem, the arrival of on-chain autonomous agents could increase liquidity and efficiency, as agents arbitrage and reinvest continuously.

VaultLayer's approach also highlights the importance of **governance and community** – by giving control via tokens (VAULTER) and transparency, it ensures that as the system grows, stakeholders can collectively steer it. This will be vital as new strategies are added or if unexpected scenarios arise; the governance process can adapt policies or code, with token holders' consensus.

In summary, **VaultLayer combines multiple frontiers**: Bitcoin to DeFi, AI to blockchain, and user intent to autonomous execution – all while maintaining the crypto ethos of decentralization and trustlessness. The combination of NFT smart vaults and Vaulter Agents could very well serve as a template for next-generation DeFi platforms. As we conclude this paper, we reiterate the core message: *secure AI delegation for NFT-owned smart vaults* is a design that opens the door to more intelligent and efficient use of digital assets.

VaultLayer invites the community to examine, collaborate, and build upon this framework – whether by writing new Lit Actions for more strategies, integrating other oracle signals, or spinning up user-friendly front-ends that simplify interacting with Vaulter Agents. By doing so, we can collectively move closer to a future where managing crypto assets is as effortless as setting an autopilot, and as safe as cryptography can make it.

References

- Lit Protocol Whitepaper (2024) – Threshold key management and Lit Actions[12][3].
- Walters et al., *"Eliza: A Web3 Friendly AI Agent Operating System,"* arXiv 2501.06781 (2025) – Integrating LLM-driven agents with blockchain interactions[4].
- Allora Whitepaper (2024) – Decentralized AI network providing price prediction feeds and inference markets[13][52].
- Rivalz Blog (2025) – Allora AI price forecasts for on-chain DeFi intelligence[33][34].
- EIP-6551: Non-fungible Token Bound Accounts (2023) – NFT-controlled smart contract wallets on Ethereum[8].
- EIP-4626: Tokenized Vault Standard (2022) – Standard for yield-bearing vault tokens, as implemented by vltCORE[45].
- Threshold Network tBTC v2 Docs (2023) – Decentralized Bitcoin-Ethereum bridge using threshold ECDSA[5].
- VaultLayer Documentation (2024) – Smart Vaults design, VaulterBTC agent, and dual-staking vltCORE protocol[1][28][44].
- Messari Report: *"State of Lit Protocol Q4 2024"* – Overview of Lit's infrastructure and VaultLayer integration[2][62].
- VaultLayer Whitepaper (draft, 2024) – CoreDAO dual staking mechanism and cross-chain interoperability approach[6][48].

---

[1] [9] [10] [12] [14] [15] [17] [18] [19] [20] [21] [22] [23] [24] [25] [54] [55] Smart Vaults | VaultLayer | Documentation

https://docs.vaultlayer.xyz/components/smart-vaults

[2] [3] [31] [32] [35] [62] State of Lit Protocol Q4 2024 | Messari

https://messari.io/report/state-of-lit-protocol-q4-2024

[4] [2501.06781] Eliza: A Web3 friendly AI Agent Operating System

https://arxiv.org/abs/2501.06781

[5] [11] tBTC Bitcoin Bridge | Threshold Docs

https://docs.threshold.network/applications/tbtc-v2

[6] [7] [16] [26] [47] [48] [50] [51] Whitepaper.md

file://file-EtUkGfEFN2QftZ2YCxUMFR

[8] How to Create and Deploy a Token Bound Account (ERC-6551) | QuickNode Guides

https://www.quicknode.com/guides/ethereum-development/nfts/how-to-create-and-deploy-an-erc-6551-nft

[13] [52] research.assets.allora.network

https://research.assets.allora.network/allora.0x10001.pdf

[27] [44] [45] [46] [49] [58] [61] vltCORE | VaultLayer | Documentation

https://docs.vaultlayer.xyz/components/vltcore

[28] [29] [36] [37] [38] [41] [42] [43] [57] VaulterBTC AI Agent | VaultLayer | Documentation

https://docs.vaultlayer.xyz/components/vaulterbtc-ai-agent

[30] [56] Intro to VaultLayer | VaultLayer | Documentation

https://docs.vaultlayer.xyz/

[33] [34] [39] [40] Rivalz AI Oracles Meet Allora Inferences: Unleashing Real-Time On-Chain Intelligence

https://blog.rivalz.ai/rivalz-ai-oracles-meet-allora-inferences-unleashing-real-time-on-chain-intelligence/

[53] Introducing Morpho-Aave-V3: An improved lending experience

https://morpho.org/blog/introducing-morpho-aave-v3-an-improved-lending-experience/

[59] What is the Threshold Network? | Threshold Docs

https://docs.threshold.network/

[60] Tokenomics | VaultLayer | Documentation

https://docs.vaultlayer.xyz/roadmap/tokenomics