SatLayer: Multi-Chain Security with BTC Restaking and Bitcoin-Validated Services

SatLayer Core Team

May 18, 2025

Abstract

We propose SatLayer, a Bitcoin restaking protocol designed to deliver robust, versatile functionality for decentralized applications. SatLayer is implemented as a layer on top of Babylon, a Bitcoin-backed Proof-of-Stake ledger, and allows developers of decentralized applications to directly tap into Bitcoin's capital. SatLayer introduces the concept of Bitcoin-Validated Services (BVS), which are applications economically secured by Bitcoin. In this way, SatLayer expands Bitcoin's established security and liquidity through "restaking," a novel security paradigm previously prominent within Ethereum ecosystems. SatLayer enables Babylon stakeholders to opt in to secure BVSs in an accountable manner, such that if a stakeholder who opts in to secure a BVS misbehaves, in a slashable manner well-defined by the BVS, then the stakeholder's bitcoins are slashed. In general, the level of security of a BVS is directly dependent on the amount of restaked bitcoins, where more bitcoins lead to greater economic security. In this paper, we describe the high level architecture of SatLayer, including the various smart contracts via which it is implemented, the interactions between the different types of parties that act in the SatLayer ecosystem, and the core security and economic properties that SatLayer aims to guarantee.

1 Introduction

Bitcoin [Nak08] launched a new paradigm of financial services, based on decentralized infrastructure. To this day, Bitcoin remains the leading cryptocurrency, in terms of market capitalization, and is underpinned by a robust blockchain-based distributed ledger. However, despite its resilience and strong security guarantees, Bitcoin's limited programmability restricts the deployment of generic applications. Although this limitation was overcome by systems that enable smart contract programming, such as Ethereum [W+14], these systems' economic resilience is typically far weaker than Bitcoin's.[1]

Recent years have seen increased efforts that aim to tap into Bitcoin's significant, yet mostly idle, capital. The main ideas are to use Bitcoin's capital in (i) DeFi applications, and (ii) to provide full smart contract capabilities backed by Bitcoin Security guarantees. In particular, Babylon [TTG+23, Tea23] allows

[1] For reference, as of March 2025, Ethereum was the second largest cryptocurrency, with market capitalization that exceeded $230B, at a time when Bitcoin's market capitalization was $1.6T. (Source: CoinMarketCap)

Bitcoin owners to secure any Proof-of-Stake (PoS) chain against forking in a fully trust-minimized and non-custodial way. Importantly, the participants are fully accountable for their actions and their staked bitcoins are slashable if the participants misbehave by signing two blocks at the same height. However, double signing protection is the only Bitcoin-backed security guarantee that Babylon provides. Thus, many applications such as oracles, lending markets, decentralized exchanges, insurance applications, and others cannot be secured by Babylon.

A practical solution to this problem is "restaking" protocols. These protocols offer the ability to maintainers of the consensus layer, that is the participants that stake their assets in order to participate in PoS, to opt in to validating applications that are built on the ledger. In this setting, the participants grant the ability to the restaking protocol to impose more nuanced slashing conditions, which are implemented by the application that the participants opt in to secure. In other words, participants are now accountable for their actions both in running the consensus protocol and in maintaining the decentralized application that is supported by the restaking protocol. If a participant misbehaves in either of the two domains, they are penalized by having their stake slashed.

Distributed ledgers that use PoS, either native (Ethereum) or via external staking (Babylon), offer the infrastructure, on top of which decentralized applications can be built. Notably, although the infrastructure itself is secured by the ledger token's capital, the applications themselves only inherit the ledger's security, but not economic, guarantees. For example, various decentralized applications that offer lending, oracles, or similar services implement their own tokens in order to provide incentives for their users, manage governance, etc. In these cases, even if the ledger on top of which an application is built is economically secure, the application itself may not have such guarantees, e.g., if its own token's capitalization is low. This issue is particularly significant for bootstrapping new applications, that is keeping them secure and stable long enough to accumulate enough capital to support their own tokens.

Combining the above two ideas, i.e., enabling restaking on a Bitcoin-backed PoS chain, is a powerful tool. SatLayer addresses exactly this concern by building a restaking layer on top of Babylon. SatLayer enables Bitcoin to secure an arbitrary set of decentralized applications by enabling programmatic slashing of restaked Bitcoin collateral.

Contributions

BTC as the Security Asset. The main contribution of SatLayer is utilizing Bitcoin as its foundational cryptoeconomic security asset. Contrary to most restaking protocols that depend on Ethereum, SatLayer enables Bitcoin holders to actively participate in the maintenance and security of decentralized applications, while also generating yields and further reinforcing the network's trust model. Through Bitcoin Validated Services (BVSs), SatLayer strengthens the security of decentralized applications by rooting validations in Bitcoin's stability and security, effectively reducing reliance on centralized trust assumptions and/or cryptocurrency assets with less capital than Bitcoin.

Cross-Chain Compatibility. SatLayer integrates seamlessly with multiple blockchain ecosystems, including Ethereum Virtual Machine (EVM), Solana

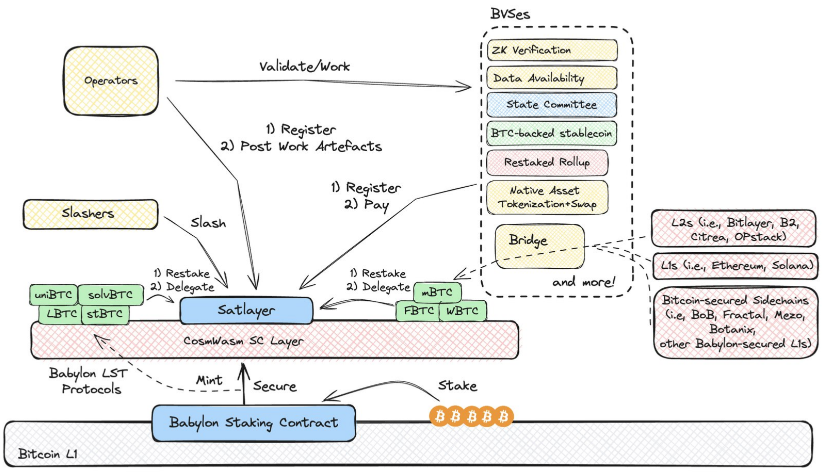

Figure 1: Diagram of the different roles in SatLayer's V1 ecosystem. SatLayer itself inherits the security guarantess of Bitcoin through Babylon Genesis, and provides a platform for tokenized BTC to be restaked and used to secure arbitrary sets of off-chain services called Bitcoin Validated Services (BVSs). Operators validate these BVSs and receive rewards for their work and can be punished via slashing if they misbehave.

Virtual Machine (SVM), and MoveVM. Therefore, it reduces the architectural complexity of migrating to its system and enhances user and developer experience across different blockchain ecosystems. In practice, SatLayer's design is centered on Babylon, where its main contracts are deployed, while also having "client" smart contracts deployed on external blockchains. These clients communicate with the core (Babylon) contracts via reliable communication protocols, which ensure dependable cross-chain interactions, secure data transfer, and low latency.

Note: Cross-chain compatibility is a feature that will be implemented in V2 of SatLayer. In the first version, all assets and logic will be hosted and implemented on Babylon.

2 The SatLayer Framework

SatLayer expands Bitcoin's value proposition by connecting its native cryptoeconomic guarantees to the security of external decentralized applications. In this section we outline the different components that form SatLayer's ecosystem and infrastructure.

2.1 Participant Roles

The SatLayer ecosystem involves three key participant roles: (i) BVS developers; (ii) Bitcoin restakers; (iii) node operators. Naturally, a party may act in one or more of these roles at the same time.

BVS developers

BVS developers create the decentralized services that are secured via SatLayer's Bitcoin restaking functionality. These services are essentially a combination of off-chain node sofware and smart contract-based Dapps (decentralized applications) that are deployed on top of a blockchain supported by SatLayer.[2] Consequently, BVS developers can launch their applications on a blockchain that suits their functionality and performance needs and enhance their security with Bitcoin staking via SatLayer.

Bitcoin Restakers

Bitcoin restakers are parties that own bitcoins, or Bitcoin-pegged assets, and want to use them as restaking collateral in SatLayer. Restakers participate by depositing their tokens into SatLayer's smart contracts. Information about restaked collateral assets and balances across all chains is relayed to the Babylon-based core contracts which coordinate the protocol's operation. In practice, this data trasnfer is done by bridging the deposit information from the (chain that hosts the) client contract to the core contracts. The system enables the Bitcoin restakers to delegate their stake to operators (see below). These restakers are incentivized by receiving rewards in exchange for the usage of their tokens in securing BVSs. However, they also undertake the risk of operating correctly, since misbehavior in a provable manner can lead to their tokens getting slashed by the SatLayer protocol. For this reason, it is crucial to delegate the stake to a trustworthy and reliable operator.

Node Operators

Node operators participate in the maintenance of BVSs, manage off-chain computations, ensure security, and fulfill operational responsibilities. In essence, node operators are responsible for executing the off-chain logic of BVSs, maintaining their on-chain state, and ensuring the correct execution of the BVS, including submitting proofs of misbehavior that enforce slashing of deviating operators. Operators can have stake delegated to them by Bitcoin restakers, which represents their weight in the system as well as their trustworthiness. Operators with more stake may weigh heavier when it comes to decision-making, e.g., if a BVS requires a majority of its operators to agree on a certain operation. An operator with more stake also may stand to lose more in case of misbehavior that leads to slashing. Therefore, Bitcoin restakers may opt to delegate their stake to operators that also contribute high amounts of stake on their own, i.e., have more "skin in the game".

2.2 Architecture

SatLayer employs a modular architecture in order to separate restaking, slashing, reward distribution, and BVS-specific logic. Briefly, SatLayer's core logic is implemented in WebAssembly (WASM)-based smart contracts on Babylon Genesis. This enables bitcoins which are staked to secure Babylon Genesis to

[2] SatLayer's first version will only support Babylon, whereas version 2 will support additional blockchains like Ethereum, Sui, Solana, etc.

be restaked as Liquid Staked Tokens (LSTs) in SatLayer. SatLayer's specialized smart contracts on external blockchains act as clients, enabling cross-chain data transfers. Finally, SatLayer operators run nodes, in order to manage data processing, event listening, and on-chain state of BVSes.

In more detail, SatLayer's infrastructure comprises the following components.

Smart Contracts

SatLayer's contracts consist of the core contracts, deployed on Babylon Genesis, and "client" contracts deployed on external blockchains.

The core contracts are implemented in Cosmos WebAssembly (CosmWasm) and coordinate the interactions between BVS applications, node operators, and restakers. In essence, the primary role of these contracts is managing the staking and slashing of the bitcoins that support the system.

The client contracts are deployed on blockchains like Ethereum, Sui, etc. These specialized contracts handle the deposits of Bitcoin-pegged assets in SatLayer and the interactions with BVSs deployed on these blockchains. For example, a party that has Bitcoin-pegged assets on Ethereum, such as wrapped BTC, can deposit their assets on the SatLayer client contract that is deployed on Ethereum. Data regarding these deposits is relayed to SatLayer's core contracts, which are deployed on Babylon, so that the party can delegate their restaked collateral to operators. If the operator that manages this stake misbehaves in a provable manner, then the assets, which are staked on the Ethereum client contract, are slashed. Otherwise, they party can withdraw these assets after a period of time, alongside possible rewards that they have accrued over time.

Bitcoin Validated Service (BVS)

The introduction of Bitcoin Validated Services (BVSs) are the major contribution of SatLayer. A BVS is a decentralized service that is *accountably* secured by Bitcoin stake. In essence, the cryptoeconomic security of a BVS is supported by the value of Bitcoin and increases as more Bitcoin participates in the BVS's operation. BVSs are deployed by developers either on Babylon Genesis or external blockchains which are supported by SatLayer's client contracts. A BVS defines the application logic of the service, including, crucially, specific slashing conditions. These conditions describe which deviations by the service's operators should be penalized.

For example, consider a BVS that implements a data oracle which is deployed on Babylon Genesis and provides information about Ethereum-based events. This BVS is maintained by a set of SatLayer operators, who are responsible for providing Ethereum data (i.e., price feeds) to interested parties. If an operator misbehaves, e.g., by providing incorrect data , then they should be penalized for this deviation. In our example, the BVS should define how this misbehavior can be identified and when the penalization, that is the slashing of the bad operator's stake, is enforced. For instance, the BVS could require operators or service managers to regularly submit proofs of data existence (or non-existence). In this case, these proofs would need to be resolved on Babylon Genesis (i.e., via an Ethereum light client) and subsequently used to initiate slashing events or distribute rewards to operators. Alternatively, the BVS could require that a majority of its operators support the event's submission or, contrarily, its dispute; in that case, rewards and penalization can be enforced as long as this majority of operators is honest.

Operators

The final element of SatLayer concerns the operators. Operators run specialized software for each BVS they validate that monitors on-chain and off-chain events and executes necessary computations for BVS functioning. In essence, the correctness of the operator computation is backed by the restaked Bitcoin delegated to those operators. Importantly, these components may interact with multiple different blockchains, e.g., to monitor events in one and submit data to another.

Following the above oracle BVS example, an operator would need to connect to both Ethereum and Babylon. Specifically, the operator needs to listen for the Ethereum events corresponding to the data the oracle provides. After listening to such event, the operator should submit it to the (Babylon-based) BVS contract. Consequently, the operator needs to operate (or connect to) an Ethereum full node, in order to observe the chain and collect the events, as well as operate a Babylon client, in order to submit new events. Additionally, the operator may need to transform the event in a manner compliant with the BVS's requirements. All of these operations, from the event collection to its processing and then its submission, are conducted off-chain.

2.3 Operational Workflow

SatLayer defines clear and structured workflows involving all ecosystem components and participants. Following, we present the workflows of the following core SatLayer operations: (i) vault creation and registration (ii) BVS maintenance and execution (iii) BVS rewards and slashing

Vault Creation and Registration

Bitcoin restaking via SatLayer comprises of the following main stages (cf. **??**).

Interested node operators must first deploy a vault via the BVS-vault-factory [3] and register in the BVS-registry [4]. The address which the operator used to deploy the vault is registered in the BVS-registry. The vault itself enables Bitcoin holders to deposit Bitcoin collateral, effectively restaking it and delegating it to the owning operator. The BVS registry is responsible for maintaining the records of operators and BVSs and which ones are connected to each other.

BVSs which want to make use of Bitcoin security must also register their onchian footprint on the BVS-registry. In essence, a BVS registers an EOA address that it controls in the bvs-registry, then registers the operator they wish to work with. The operator then opts into the BVS in the registry, creating bilateral opt-in. Both the operator and the BVS can choose to terminate the relationshio at any time [5]. the on-chain interactions of the BVSE. represents a supported token (e.g., Bitcoin, native Babylon or other supported tokens) used to validate one

[3] `https://github.com/satlayer/satlayer-bvs/tree/main/crates/bvs-vault-factory`

[4] `https://github.com/satlayer/satlayer-bvs/tree/main/crates/bvs-registry`

[5] but the unbonding time must be large enough to prevent economic attacks

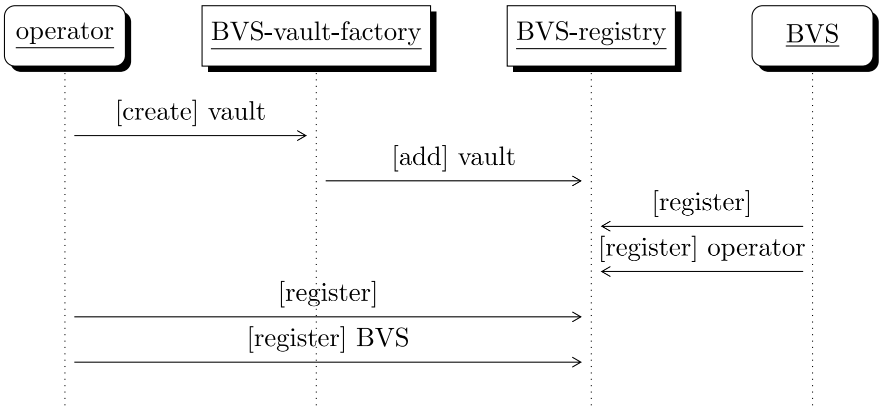

Figure 2: Workflow of initializing an operator and registering an operator to validate a Bitcoin Validated Service (BVS).

or more BVSs. Note that if the tokens are BTC-pegged assets on non-Babylon chains, then these assets are controlled by the Babylon-hosted SatLayer core contracts via secure bridging protocols.

Every BVS-vault's stake is delegated to the operator which owns that vault. Bitcoin holders wishing to generate yield from bitcoin restaking simply have to deposit to the vault of their chosen operatorIn effect, users delegates their strategy shares to registered operators and receive rewards, corresponding to their delegate's performance, or get slashed if the delegate misbehaves.

At this point the operator can choose how to distribute their stake (both own and delegated) among running Bitcoin Validated Services (BVSs). We will describe in more detail the registration and execution of BVSs in the following paragraphs.

Finally, when a party no longer wants to participate in staking, they withdraw their deposited assets, along with any rewards that they accrued from their participation in SatLayer. This is done by calling the relevant withdrawal operations of the BVS-vault contract.

Finally, bitcoin holders who which to generate yield on their BTC can deposit it in

BVS Execution and Maintenance

A major requirement for SatLayer's ecosystem success is a thriving and diversified set of Bitcoin Validated Services (BVSs). The services are deployed as a combination of off- and on-chain components on any of the chains supported by SatLayer; for the purposes of this description, we assume that it is deployed on Babylon (cf. Figure 3).

Once an operator and a BVS have mutually opted in, a feedback loop emerges where each member of the system performs specific tasks in order to keep the BVS functioning properly. The distnction between this and other cryptoeconomic systems is that the operator is guaranteeed to behave honestly due to the Bitcoin that has been restaked it.

The main task of node operators is executing the off-chain logic of a BVS, submitting the results of the relevant tasks, and validating the correctness of other operations. BVS developers are free to define the output of their tasks as they deem fit.

The execution of the BVS's logic is done via tasks, which are submitted by the BVS's clients. Each operator of the service monitors the BVS for newly-submitted tasks. When a new task is submitted, as a function call to the BVS contract, the operator collects the task's parameters and performs the necessary computations. The results of these computations are collected by the aggregator (off-chain) module, which publishes them on the BVS contract. Some BVSs may have the operators directly submit results and forego the aggregator module.

Finally, the BVS is responsible for defining the validation of the submitted results, based on well-defined correctness criteria. For example, it might require a submission of a non-interactive short proof of execution correctness (e.g., SNARK). Alternatively, it could require all operators to input the same result and, in case of dispute, perform a bisection game between the two disagreeing operators to identify which of them misbehaved and in which exact step of the operation.[6] In the most straightforward case, the BVS can assume that a majority of its operators (e.g., weighted by stake) is honest and accept the result that the majority agrees upon. Naturally, each of these alternatives comes with different correctness assumptions and guarantees, so both the BVS developers and operators should carefully explore which approach is the best for each service.

Rewards and Slashing

Users that participate in the SatLayer protocol, either directly as node operators or by delegating their bitcoins, earn restaking rewards. The calculation and distribution of these rewards is the responsibility of the BVS and proceeds as follows (Figure 4).

After the completion of one or more operations, the BVS's manager computes the amount of rewards that should be given to each participating node operator. The rewards are formatted as a Merkle tree, which encodes how many tokens each address should receive. Following, the BVS's manager submits the Merkle tree's root to the BVS-rewards contract, along with the total amount of rewards that will be distributed.

Eventually, each node operator can claim their rewards by calling the BVS-rewards contract. Specifically, they submit a Merkle inclusion proof for their address and amount of rewards that need to be claimed. The contract validates the proof with respect to the submitted Merkle tree's root and then sends the claimed rewards to the operator.

Note: The BVS itself is trusted to disperse rewards correctly. In particular, it is trusted to compute each operator's and stakers' rewards, construct the Merkle tree, and submit its root on chain.

Finally, satlayer enables accountability of operators in the form of *slashing.*

[6] Running a bisection game assumes that the task's operations can be expressed as an ordered list of deterministic sub-tasks, each of which can be easily verified by the contract. In this case, each operator can submit the intermediate states (after performing each sub-task) in a Merkle tree and, in case of disagreement, the two parties find the first leaf (that is sub-task) of difference and validate which of the two is correct.

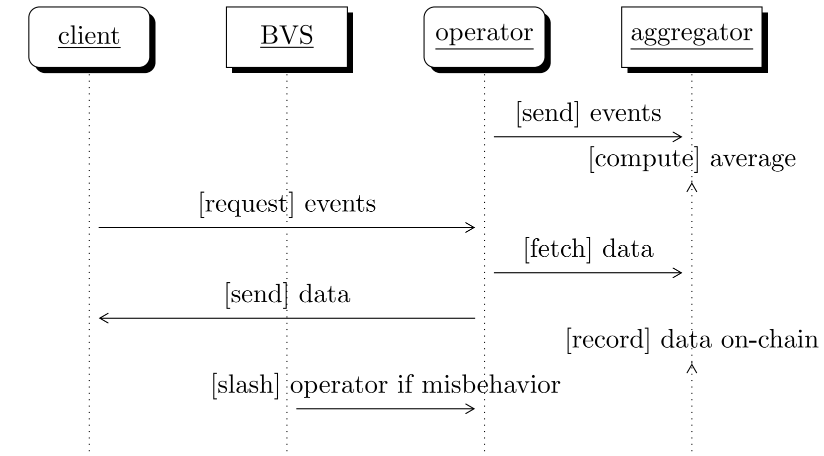

Figure 3: Example workflow of a BVS that implements an Ethereum oracle, where the user requests the events emitted by a given Ethereum smart contract. The operator fetches the events from anaggregator node as well as provding its latest measurements. The aggregate data is returned to the user, and the individual measurements are stored so that operators can be slashed if they misbehave.

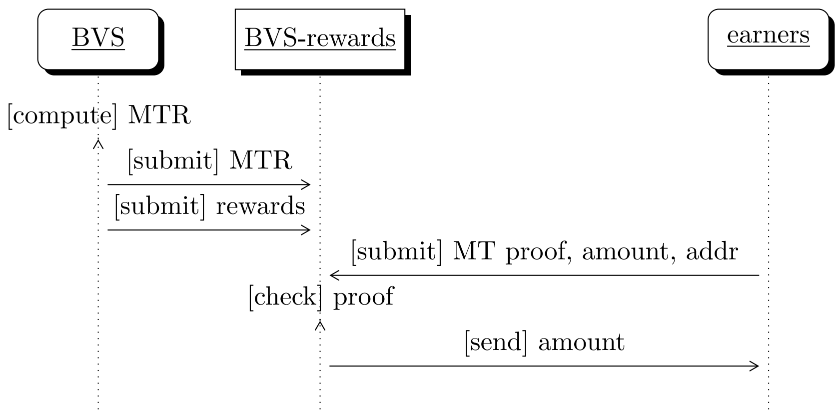

Figure 4: Workflow of rewarding earners (node operators and restakers).

in particular, when an operator misbehaves, they may be penalized by forfeiting part or all of their assets, as follows.

To enable slashing for thier application, a bvs must first call 'enable-slashing' on the bvs-registry contract with the desired slashing parameters. the operator must also opt into the slashing by calling 'operator-opt-in-to-slashing'.

A request for slashing is initiated by the slash initiator, which is the same entity as the bvs. The on-chain system used to initiate the slash can be as simple as an eoa or could be a smart contract system. the slasher sends a request slashing message to the bvs-router using the address that they registered their service with. after the request slash is sent, the operator has a grace period to submit a 'cancel-slash' request. the slash intiator can choose to cancel the slash if the operator has addressed the issue (in a way defined by the service). if the operator fails to respond or responds unsatisfactorily, their funds will then be moved to a locked state. the slashed funds are then routed appropriately in a final step to those who are entitled to them.

3 Incentives

Designing a reward mechanism that aligns user and developer incentives with the intended functionality is crucial for the secure operation of the protocol. In this section we outline SatLayer's fee and reward mechanism and how rational participants are incentivized to act honestly.

3.1 Reward Program

The first option for getting rewarded is SatLayer's Reward Program.[7] Briefly, the program's goal is to attract Bitcoin liquidity to the system during its early stages. For this reason, early users that restake their bitcoins on SatLayer earn rewards in the form of SatLayer's native reward, Sats^2.[8]

3.2 BVS Fees and Operator rewards

Deploying a BVS and using it incurs certain fees, which are awarded to the maintainers of the BVS. Each BVS defines its own reward conditions and methods for calculating distribution amounts, as well as how these rewards are distributed, e.g., in the form of the BVS's native tokens or other such assets. This freedom leads to a market for attracting BVS operators, since these participants are expected to opt for validating BVSes with a lower cost-to-reward ratio.

Example 1: Data oracle In Section 2 we described a BVS example of a data oracle. Such services typically charge fees to the users in the form of their native tokens.[9] These fees are given to the service's operators, that is the nodes that perform the data collection and migration from the source to the destination. In the above example, the data are events, the source is the

[7] `https://docs.satlayer.xyz/partners/satlayer-rewards-for-partners`

[8] The SatLayer Reward's program will run prior to the launch of the full protocol. More information about the program can be found at `https://docs.satlayer.xyz/restakers/sats-staker-rewards`.

[9] For example, Chainlink charges its users in its token (LINK) to send function requests. (`https://docs.chain.link/chainlink-functions/resources/billing`)

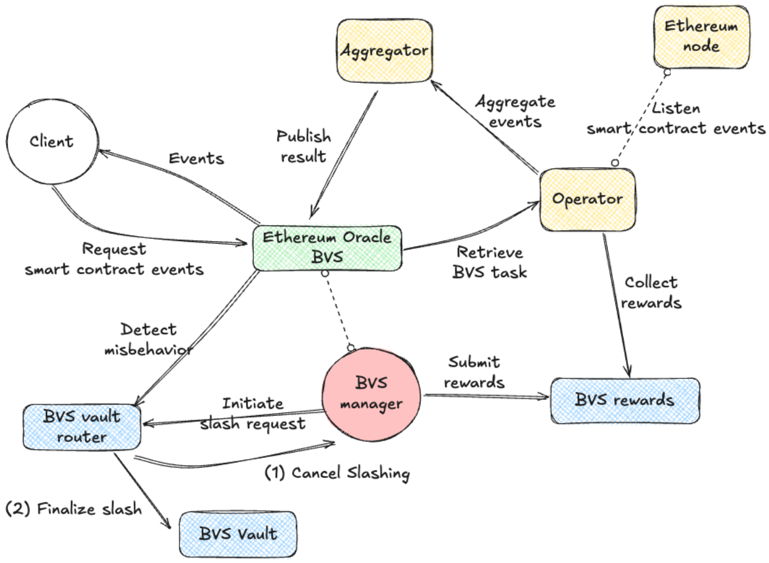

Figure 5: Example workflow of a BVS that implements an Ethereum oracle. Note that blue nodes correspond to Satlayer-related smart contracts hosted on Babylon, whereas yellow nodes correspond to off-chain components that are run by the BVS's operator. The "Ethereum Oracle" BVS is also a smart contract hosted on Babylon, whereas the client and BVS manager are off-chain parties.

Ethereum blockchain, and the destinations are the BVS (for record-keeping) and any clients who request the data. When using SatLayer, the BVS operators would again be paid in the BVS's native token for performing their duties. Additionally, these operators would have BTC restaked and delegated to them. Therefore, if a misbehavior occurs, the misbehaving operator would have its bitcoins slashed. In effect, the positive incentive for operators is the same as traditional services, while the counter-incentive (against misbehavior) is rooted on Bitcoin, instead of relying on the service's native token which, especially in its early stages, may be volatile and not as valuable.

Figure 5 depicts the workflow of this example BVS of a data oracle for events emitted by an Ethereum-hosted smart contract. Briefly, first a client submits a request for retrieving events from the Ethereum smart contract. The BVS collects such task requests and the BVS's operators retrieve them and execute them. To do so, an operator connects to an Ethereum node, in order to retrieve the events, and operates an aggregator, in order to collect the events and submit them to the BVS. After the task is finished, e.g., when a majority of operators agree on the same response, the BVS responds to the client. In addition, the BVS's manager is responsible for collecting the operator responses and computing the reward allocation appropriately. Finally, a misbehavior may be detected, in this example by the BVS manager and/or the BVS itself. In this case, the slashing process is initiated and resolved, either being rejected (vetoed by the resolver) or accepted (and enforced on the BVS vault).

Example 2: Decentralized Exchange (DEX) Another BVS example is a Bitcoin-native DEX. In this case, users can trade their tokens (e.g., BRC-20[10]) on the SatLayer-backed DEX, with trades being settled directly on Bitcoin. Typically, DEXs charge fees to users for each trade, which are then distributed to the DEX's liquidity providers, i.e., users who deposit their tokens to the DEX and enable trades. In our case, part of these fees would be given to the BVS operators, in exchange for correctly settling transactions. Importantly, if a DEX operator misbehaves, e.g., trying to settle an incorrect trade in an identifiable manner well-defined by the DEX, then their restaked bitcoins would be slashed. Therefore, the BVS operators would stand to lose not only the BVS rewards, but also their restaked bitcoins. In addition, the DEX's security (regarding settlements) would inherit the security properties of the Bitcoin blockchain.

3.3 Slashed Assets

The final element of the incentives' mechanism concerns slashing.

The first concern in terms of slashing is the level of slashed assets. Specifically, the protocol defines whether slashing of a misbehaving party is *partial*, where only part of their assets are forfeited, or full, where all assets of the party are confiscated. Since slashing is the primary mechanism for disincentivizing misbehavior, the choice between partial and full slashing, e.g., in the case of particularly serious offenses, should be carefully designed and implemented.

The second question revolves around the handling of slashed assets. There are typically two options in this case. First, the slashed assets may be pooled are redistributed or reused in ways that are beneficial for the protocol, e.g., via a robust governance protocol. Alternatively, the slashed assets may be burnt. The choice between the two options offers a tradeoff between enhanced functionality and opportunities (when repurposing) or implementation simplicity and avoiding side-effects (when burning), so the mechanism should be carefully designed and implemented.

4 Interoperability

In this section we outline SatLayer's interoperability characteristics.

4.1 Multi-Chain Connectivity

SatLayer aims to enable Bitcoin staking across diverse blockchain ecosystems. This is enabled via the bridging of Bitcoin assets from various chains to Babylon, followed by restaking them on SatLayer. Such cross-chain connectivity is implemented through robust communication protocols ensuring synchronization between Babylon and external chains and promotes state consistency.

SatLayer's deployment on Babylon Genesis acts as the control plane upon which activity from external chains (i.e., staking, slashing, BVS registration, etc.) is synchronized.

[10] `https://l1f.discourse.group/t/brc-20-introduction-to-brc-20/72`

4.2 Application-Level Integration

Currently, various chains support different programming languages, execution environments, virtual machines, and accounting paradigms (e.g., UTxO vs accounts). Consequently, a service which is developed under one chain's framework may need to be migrated, or even completely redesigned, in order to be redeployed to a different chain. This increases friction for projects, leading to "siloed" settings where applications cannot leave a particular ecosystem without incurring a massive overhead for the developers.

SatLayer aims to alleviate such transition from alternative "Application Validated Services (AVSs)" by building an interoperability layer that allows restakers and BVS developers to deposit capital and deploy their on-chain applications on any SatLayer-supported chain.

In essence, SatLayer will support an interoperability stack which enables data transfers between Babylon and external chains. This functionality allows developers to build and deploy their applications on any SatLayer-supported chain that they think best fits their application's requirements, while SatLayer handles the underlying communication and coordination of these applications from its Babylon-hosted core. This has two main benefits for developers. First, they are not forced to migrate or rewrite existing applications to a new environment. For example, a BVS written in Solidity can be deployed on Ethereum and be secured by SatLayer via its interoperability stack, instead of having to be rewritten in WASM. Second, they can choose which supported chain fits the needs of their application best. For example, a BVS that relies on Ethereum's RANDAO as a source of randomness could be deployed on Ethereum, which supports native access to RANDAO values, instead of being deployed to a different chain and requiring transfers of the relevant RANDAO values to that chain.

5 Security and Economic Outline

SatLayer's core proposition is based on utilizing Bitcoin as a cryptoeconomic security asset. In this section we outline the security properties that SatLayer aims to guarantee, the assumptions that may be needed to achieve this, the points that need to be carefully considered, as well as relevant notions from the literature. Note that the formal definition of these properties and rigorous analysis of SatLayer is outside the scope of this paper.

5.1 Execution Properties

Correctness

A core property that SatLayer should guarantee is the execution correctness of the BVSs supported by the restaking process. Essentially, SatLayer should guarantee that, under a realistic set of honesty and/or rationality assumptions, a BVS which is supported by SatLayer operators outputs the correct value during its execution. For example, these assumptions could include honest majority (in a Byzantine analysis) or that all participants are rational (in a game theoretic analysis).

Correctness is not restricted to the BVSs, but also applies to the SatLayer protocol itself. For example, the protocol should ensure that its main functionalities, such as registration and removal of operators, stake registration and delegation, reward distribution, and slashing enforcement, should be correctly executed.

Availability

The second major property that SatLayer should guarantee is that the supported BVSs progress over time. In essence, SatLayer's mechanism should ensure that, under a set of assumptions as above, a BVS does not stall, observe outage, or halt altogether.

As with correctness, availability should also be guaranteed for the SatLayer protocol itself, in addition to the BVSs that it supports. Specifically, SatLayer should ensure that operators and stakers can freely join or leave the system, as well as operations such as stake delegation, reward withdrawal, or recording of slashing cannot be censored.

Latency, Throughput, and Necessary Delays

The analysis of SatLayer should not only focus on security but also performance. For instance, it should outline how quickly operators and stakers can join or leave the system (latency), as well as how many BVSs can be (reasonably) secured in parallel with any given set of stakers (throughput). In addition, the analysis should outline how the composition of SatLayer with other systems introduces delays that are necessary for its correct operation. For example, if a BVS requires the bridging of data across chains, then the analysis should outline how SatLayer's latency is affected due to the need of completion of such operations. Additionally, the protocol should account for the latency of the Bitcoin blockchain, e.g., when it comes to enforcing penalties for misbehaving validators. For example, if a validator is slashed and needs to be removed from SatLayer, the performance of the system may be affected, e.g., the operation of BVSs in which the misbehaving participation operates may be delayed until the slashing is finalized on Bitcoin.

5.2 Economic Analysis

The above properties concern the execution guarantees of SatLayer. Since SatLayer's design is cryptoeconomic in nature, where security argumentation mostly revolves around Bitcoin's capital offerings, we now focus on the game theoretic and cryptoeconomic aspects of the mechanism.

Incentive Compatibility

The main focus of a game theoretic analysis should be to show that the system, which includes the protocol and the incentives' structure, is incentive compatible. In essence, the analysis would assume that all participants (cf. Section 2.1) are rational, that is they aim to maximize their utility. Typically, a party's utility is the amount of rewards that they gain from participation, possibly minus the cost of executing the protocol and/or including potential external gain from deviating from the protocol (e.g., bribes or double spending). Following, the goal is to show that, under this rationality assumption, all participants are incentivized to follow the protocol. Results of such analysis include a Nash equilibrium, where every individual party follows the protocol as long as everybody else does so, or strategy dominance, where a party follows the protocol regardless of what others do.

Note that incentive compatibility relates both to SatLayer and the BVSs themselves. In particular, if SatLayer is incentive compatible then rational operators will follow the protocol when registering a BVS, restaking, and performing protocol-related tasks like submitting slashing requests. Similarly, if a BVS is incentive compatible then rational operators will execute the tasks correctly and timely.

Operator collusion

The game theoretic analysis described above often considers parties as individuals, with each trying to maximize their own utility. However, in some cases it is reasonable to assume that parties may collude. In this case, parties form coalitions and coordinate their actions, e.g., in order to increase the coalition's aggregate utility (compared to operating on an individual basis). A goal in this case could be to show that the protocol is resilient, i.e., it is an equilibrium or dominant strategy, even in the presence of such coalitions.

Collusion principally concerns both the SatLayer and the BVS's operation. For example, if slashing is enforced via social consensus, then colluding operators may vote against a valid slashing request to avoid penalization, which should otherwise be guaranteed by SatLayer. Similarly, if some operators collude with the BVS manager, then the latter may miscompute the reward allocation in their favor.

Accountability

A major component of the game theoretic analysis of SatLayer will be exploring its accountability guarantees. Briefly, accountability encompasses how faults are attributable and the ways in which parties are penalized for misbehaving. Accountability has been widely explored in the literature, e.g., [NTT21], so here we will outline some aspects of it and leave a thorough investigation of it for future work.

A first consideration when it comes to accountability are the limits to enforcing it in a non-interactive manner. Briefly, if a non-interactive proof of misbehavior can be created, w.r.t. to a well-defined set of rules, then accountability can be enforced deterministically at the protocol level, e.g., on the ledger itself. However, faults are not always objectively attributable.[11]

When a non-interactive proof of misbehavior cannot be created, the system needs to rely on the participation, and possibly honesty, of other parties. For example, the parties may vote on whether a misbehavior occurred and, assuming an honest (super)majority, penalize the misbehaving party without needing to produce a mathematical proof. However, an additional consideration here, beyond the need for honesty assumptions, is that faults cannot always be objectively attributed to a single party, even if the supermajority of participants

[11] For example, see [Eig24, Section 1.2] for a discussion on fault categorization.

is honest. For example, a fault may occur in a way that either of two parties is at fault, without being able to pinpoint which of the two.

Furthermore, a second point of discussion concerns unintentional slashing. For example, software bugs, network outages, or other unforeseen elements may hinder a (otherwise honest) party from participating in the protocol. The accountability mechanism should take into account such unintentional faults and refrain from severely penalizing parties for them, albeit still disincentivizing lack of participation.

In summary, a game theoretic analysis of the SatLayer ecosystem should explore such considerations and outline the limits of accountability in practice. Note that although SatLayer is responsible for offering the platform on which accountability is enforced, the slashing conditions and the mechanism for enforcing it are the responsibilities of the BVSs themselves. Therefore, the analysis would outline the limits of responsibility of the SatLayer protocol and the BVS and explore the interconnection between the two.

Security of Underlying Ledger

Since SatLayer's core contracts are deployed on Babylon, they inherit its security and economic guarantees. Essentially, SatLayer depends on the Babylon ledger to offer safety and liveness. If Babylon's security is compromised and one of these properties is violated, then SatLayer could also have see operations censored (liveness hazard) or reversed (safety hazard). Albeit this is a fundamental concern, which stems from the fact that SatLayer is deployed on top of Babylon, it should be explored in terms of cryptoeconomic security. Specifically, the security of SatLayer should be analyzed as the minimum security guarantees of SatLayer itself and Babylon.

Validator Reuse and Cryptoeconomic Composition

Another broad area of analysis concerns the system's economic guarantees in the presence of other protocol with which it shares the validators. In the case of SatLayer this is an inherent feature, since restaking depends by design on the reuse of assets, in SatLayer's case bitcoins, to secure the system. However, economic composition of various systems via validator reuse can lead to exogenous shocks that, if not accounted for, could lead to breakdown of the protocol itself. Therefore, the analysis should outline the economic guarantees of the system w.r.t. its level of overcollateralization, such that if a number of operators suddenly become unavailable (e.g., due to being slashed in a different protocol in which they also participate), then the system can continue functioning properly. Such questions have been explored in the literature, e.g., [DR25], so they need to be tailored for SatLayer's needs and particularities.

Note that cryptoeconomic composition is a concern for both SatLayer and the BVSs. In essence, both SatLayer and the BVSs rely on reuse of external assets, namely Bitcoin, for their cryptoeconomic guarantees. Therefore, if an exogenous shock occurs, e.g., due to the slashing of Bitcoin stakers in other protocols, its effects could manifest both on SatLayer, e.g., if the slashed operators participate in its governance or operations like slashing, and on a BVS, if some of its operators get externally slashed.

6 Related Work

The concept of restaking has gained significant traction in recent years. Various industry projects explored different options and functionalities of restaking on different platforms, while the academic literature is analyzing the security and cryptoeconomic properties of such protocols. In this section we provide a brief, non-exhaustive but rather representative, list of such existing work.

Industry Projects

Using Bitcoin as the resource of a Proof-of-Stake (PoS) protocol has been explored and implemented in various systems. Babylon[12] is such a system, which uses Bitcoin as its staking asset, notably in an *accountable* manner, such that if a Babylon validator misbehaves then their bitcoins are slashed (on the Bitcoin blockchain) [TTG+23, Tea23]. BounceBit[13] uses a combination of its native token and Bitcoin, albeit the bitcoins are used indirectly since they need to be bridged (and wrapped) before being used on BounceBit. The functionality of these systems is orthogonal to SatLayer, since SatLayer can be deployed on any such protocol and inherit both its security guarantees and its supported staked assets.

EigenLayer[14] is a system that pioneered the notion of restaking on Ethereum. EigenLayer allows designers of smart contracts called Autonomous Verifiable Services (AVSs) to define specific correctness guarantees, such that if a (re)staker that maintains the AVS misbehaves then they face slashing of their staked assets, which can be native Ethereum, Liquid Staking Tokens (LSTs), the EIGEN token, or any ERC20 token. In effect, EigenLayer offers an analogous functionality to SatLayer, albeit limited to Ethereum (or Ethereum-based assets), whereas SatLayer focuses on Bitcoin instead.

Pell Network[15] offers a similar service to SatLayer. In particular, it implements Bitcoin restaking in an indirect manner, where Bitcoin holders deposit their tokens to Liquid Staking Derivatives' (LSD) platforms hosted on Babylon. In exchange, they receive LSDs, which are deposited on Pell and represent their (Bitcoin) deposit to the LSD protocols. Consequently, the bitcoins, which are deposited on the LSD platforms, are staked in Babylon, whereas the LSDs are (re)staked in the applications supported by Pell.

Symbiotic[16] is a generalized that allows developers to build their own restaking implementation and operator set. In particular, Symbiotic allows anyone to create a set of validators and define collateral assets. In order to participate, stakers deposit collateral and receive asset-specific tokens, which can be used as the means of restaking and securing various applications. Notably, Symbiotic operates primarily on Ethereum, while also offering some support of Bitcoin via integration with bitcoin wrapping services. In comparison, SatLayer is primarily focused on using Bitcoin directly and, more importantly, trustlessly as its main collateral asset.

Allstake[17] is a restaking protocol, which is built on the NEAR blockchain

[12]`https://babylonlabs.io`
[13]`https://www.bouncebit.io`
[14]`https://www.eigenlayer.xyz`
[15]`https://pell.network`
[16]`https://symbiotic.fi`
[17]`https://allstake.org`

and utilizes its *chain signature* functionality[18] in order to bridge assets, like BTC or ETH, which can then be used (in various combinations) to secure services. Notably, chain signatures, hence also Allstake, rely on the security of the NEAR blockchain. Notably, this includes slashing, wherein an operator that misbehaves when securing an AVS is slashed only if the NEAR chain's validators enforce the slashing, hence Allstake inherits the cryptoeconomic security of NEAR. In comparison, because SatLayer is hosted on Babylon, it relies directly on Bitcoin for its cryptoeconomic security.

Omnity[19] is another restaking protocol. It is hosted on the Internet Computer (ICP) and relies on its chain-key cryptography functionality[20] for bridging bitcoins and other assets to the ICP chain and then restaking them. However, similar to Allstake on NEAR, Omnity relies on the security and cryptoeconomic guarantees of the ICP, since slashing is enforced by the ICP operators, instead of being directly enforced on Bitcoin, as is the case of Babylon and SatLayer which is hosted on it.

Academic Literature

Restaking has been explored to large extent in the literature. Here we will provide only a brief overview of some important related topics, while we refer to [GVKT24] for a more in-depth exploration of the state of the art and the challenges of this line of research.

A core part of SatLayer is focused on slashing misbehaving parties. The necessity and usefulness of slashing has been widely explored in the literature. STAKESURE [DRK24] analyzes the cryptoeconomic safety of Proof-of-Stake (PoS) systems by comparing the profit and the cost of corrupting enough validators to launch an attack. Interestingly, it discusses how slashed funds can be allocated as insurance, such that they are used to compensate victims of attacks. In the same line of work, [BLR24] shows that slashing is not sufficient to guarantee cryptoeconomic security if message delays are not bounded, but can be used as a building block of a secure protocol in a synchronous network where honest parties are always active. These results demonstrate the necessity of both using and carefully designing the slashing mechanism, such that node operators are disincentivized to attack the system and slashed funds are used to further enhance its security.

The economics of restaking are explored further in [DR25]. This work explores the effects of composing restaking protocols with Layer 1 systems and the risks of reusing validators across multiple such protocols. In particular, it identifies the risks that arise for a protocol under stake shocks, when part of its validators population gets slashed due to attacks on external systems with which the protocol shares validators. To this end, the paper quantifies the level of overcollateralization needed, that is the excess between the cost and profits from an attack, such that a protocol can withstand sudden losses of (restaking) validators. The results of this work are orthogonal to the design of a restaking system like SatLayer, but nonetheless offer useful insights on the limits of its security and the risks that it may inherit, or introduce, to the protocols with which it shares validators.

[18] `https://docs.near.org/chain-abstraction/chain-signatures`
[19] `https://www.omnity.network`
[20] `https://internetcomputer.org/how-it-works/chain-key-technology`

7 Conclusion

This paper introduces SatLayer, a restaking protocol that enables Bitcoin holders to secure decentralized applications (Dapps). In essence, SatLayer allows the creation of Dapps which are economically secured by Bitcoin, in order to elevate the trust, flexibility, and scalability of decentralized blockchain ecosystems.

SatLayer's core functionality is hosted on Babylon, a blockchain with accountable Bitcoin staking capabilities. In turn, SatLayer inherits Babylon's underlying security guarantees and enables users who stake their bitcoins, either directly on Babylon or in BTC-pegged tokens hosted on external blockchains, to restake their assets in order to maintain Bitcoin Validated Services (BVSs). BVSs are smart contracts who require off-chain computations, such as operating as an oracle of off-chain events, performing computationally heavy operations, etc.

SatLayer also defines reward and slashing mechanisms. In the reward scheme, fees paid by the BVS developers and users are awarded to the BVS maintainers in order to incentivize correct and honest participation. For slashing, BVSes can specify programmatic conditions under which their operators can be slashed for misbehavior.

Finally, SatLayer enables interoperability in multiple ways. First, it allows the deployment of a BVS in any of the supported blockchains and uses a secure data transfer mechanism to coordinate these BVSs via the (Babylon-hosted) core contracts. Second, it enables the participation of holders of Bitcoin-pegged assets which live on supported blockchains, e.g., wrapped Bitcoins or alternative DeFi liquid Bitcoin-pegged tokens.

References

[BLR24] Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. The economic limits of permissionless consensus. In Dirk Bergemann, Robert Kleinberg, and Daniela Sabán, editors, *Proceedings of the 25th ACM Conference on Economics and Computation, EC 2024, New Haven, CT, USA, July 8-11, 2024*, pages 704–731. ACM, 2024.

[DR25] Naveen Durvasula and Tim Roughgarden. Robust restaking networks. In Raghu Meka, editor, *16th Innovations in Theoretical Computer Science Conference, ITCS 2025, January 7-10, 2025, Columbia University, New York, NY, USA*, volume 325 of *LIPIcs*, pages 48:1–48:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[DRK24] Soubhik Deb, Robert Raynor, and Sreeram Kannan. STAKESURE: proof of stake mechanisms with strong cryptoeconomic safety. *CoRR*, abs/2401.05797, 2024.

[Eig24] EigenLabs. Eigen: The universal intersubjective work token. `https://docs.eigenlayer.xyz/assets/files/EIGEN_Token_Whitepaper-0df8e17b7efa052fd2a22e1ade9c6f69.pdf`, 2024.

[GVKT24] Krzysztof Gogol, Yaron Velner, Benjamin Kraner, and Claudio Tessone. Sok: Liquid staking tokens (lsts) and emerging trends in restaking. *arXiv preprint arXiv:2404.00644*, 2024.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[NTT21] Joachim Neu, Ertem Nusret Tas, and David Tse. The availability-accountability dilemma and its resolution via accountability gadgets. *CoRR*, abs/2105.06075, 2021.

[Tea23] The Babylon Team. Bitcoin staking: Unlocking 21m bitcoins to secure the proof-of-stake economy. `https://docs.babylonlabs.io/papers/btc_staking_litepaper.pdf`, 2023.

[TTG+23] Ertem Nusret Tas, David Tse, Fangyu Gai, Sreeram Kannan, Mohammad Ali Maddah-Ali, and Fisher Yu. Bitcoin-enhanced proof-of-stake security: Possibilities and impossibilities. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 126–145. IEEE, 2023.

[W+14] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. `https://ethereum.github.io/yellowpaper/paper.pdf`, 2014.